Inference

Introduction

The R markdown document for this section is available here[24].

This chapter introduces the statistical concepts necessary to understand p-values and confidence intervals. These terms are ubiquitous in the life science literature. Let's use this paper[25] as an example.

Note that the abstract has this statement:

> "Body weight was higher in mice fed the high-fat diet already after the first week, due to higher dietary intake in combination with lower metabolic efficiency."

To support this claim they provide the following in the results section:

> "Already during the first week after introduction of high-fat diet, body weight increased significantly more in the high-fat diet-fed mice ($+ 1.6 \pm 0.1$ g) than in the normal diet-fed mice ($+ 0.2 \pm 0.1$ g; $P < 0.001$)."

What does $P < 0.001$ mean? What are the $\pm$ included? We will learn what this means and learn to compute these values in R. The first step is to understand random variables. To do this, we will use data from a mouse database (provided by Karen Svenson via Gary Churchill and Dan Gatti and partially funded by P50 GM070683). We will import the data into R and explain random variables and null distributions using R programming.

If you already downloaded the `femaleMiceWeights` file into your working directory, you can read it into R with just one line:

```
dat <- read.csv("femaleMiceWeights.csv")
```

Our first look at data

We are interested in determining if following a given diet makes mice heavier after several weeks. This data was produced by ordering 24 mice from The Jackson Lab and randomly assigning either chow or high fat (hf) diet. After several weeks, the scientists weighed each mice and obtained this data (`head` just shows us the first 6 rows):

[24] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

[25] http://diabetes.diabetesjournals.org/content/53/suppl_3/S215.full]

```
head(dat)
```

```
##   Diet Bodyweight
## 1 chow      21.51
## 2 chow      28.14
## 3 chow      24.04
## 4 chow      23.45
## 5 chow      23.68
## 6 chow      19.79
```

In RStudio, you can view the entire dataset with:

```
View(dat)
```

So are the hf mice heavier? Mouse 24 at 20.73 grams is one the lightest mice, while Mouse 21 at 34.02 grams is one of the heaviest. Both are on the hf diet. Just from looking at the data, we see there is *variability*. Claims such as the one above usually refer to the averages. So let's look at the average of each group:

```
library(dplyr)
control <- filter(dat,Diet=="chow") %>% select(Bodyweight) %>% unlist
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight) %>% unlist
print( mean(treatment) )
```

```
## [1] 26.83417
```

```
print( mean(control) )
```

```
## [1] 23.81333
```

```
obsdiff <- mean(treatment) - mean(control)
print(obsdiff)
```

```
## [1] 3.020833
```

So the hf diet mice are about 10% heavier. Are we done? Why do we need p-values and confidence intervals? The reason is that these averages are random variables. They can take many values.

If we repeat the experiment, we obtain 24 new mice from The Jackson Laboratory and, after randomly assigning them to each diet, we get a different mean. Every time we repeat this experiment, we get a different value. We call this type of quantity a *random variable*.

Random Variables

The R markdown document for this section is available here[26].

Let's explore random variables further. Imagine that we actually have the weight of all control female mice and can upload them to R. In Statistics, we refer to this as *the population*. These are all the control mice available from which we sampled 24. Note that in practice we do not have access to the population. We have a special data set that we are using here to illustrate concepts.

Read in the data either from your home directory or from dagdata:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/femaleControlsPopulation.csv"
filename <- "femaleControlsPopulation.csv"
if (!file.exists(filename)) download(url,destfile=filename)
population <- read.csv(filename)
population <- unlist(population) # turn it into a numeric
```

Now let's sample 12 mice three times and see how the average changes.

```
control <- sample(population,12)
mean(control)
```

```
## [1] 24.11333
```

```
control <- sample(population,12)
mean(control)
```

[26] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

```
## [1] 24.40667
```

```
control <- sample(population,12)
mean(control)
```

```
## [1] 23.84
```

Note how the average varies. We can continue to do this repeatedly and start learning something about the distribution of this random variable.

The Null Hypothesis

The R markdown document for this section is available here[27].

Now let's go back to our average difference of `obsdiff`. As scientists we need to be skeptics. How do we know that this `obsdiff` is due to the diet? What happens if we give all 24 mice the same diet? Will we see a difference this big? Statisticians refer to this scenario as the *null hypothesis*. The name "null" is used to remind us that we are acting as skeptics: we give credence to the possibility that there is no difference.

Because we have access to the population, we can actually observe as many values as we want of the difference of the averages when the diet has no effect. We can do this by randomly sampling 24 control mice, giving them the same diet, and then recording the difference in mean between two randomly split groups of 12 and 12. Here is this process written in R code:

```
##12 control mice
control <- sample(population,12)
##another 12 control mice that we act as if they were not
treatment <- sample(population,12)
print(mean(treatment) - mean(control))
```

```
## [1] 0.5575
```

Now let's do it 10,000 times. We will use a "for-loop", an operation that lets us automate this (a simpler approach that, we will learn later, is to use `replicate`).

[27] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

```
n <- 10000
null <- vector("numeric",n)
for (i in 1:n) {
  control <- sample(population,12)
  treatment <- sample(population,12)
  null[i] <- mean(treatment) - mean(control)
}
```

The values in `null` form what we call the *null distribution*. We will define this more formally below.

So what percent of the 10,000 are bigger than `obsdiff`?

```
mean(null >= obsdiff)
```

```
## [1] 0.0138
```

Only a small percent of the 10,000 simulations. As skeptics what do we conclude? When there is no diet effect, we see a difference as big as the one we observed only 1.5% of the time. This is what is known as a p-value, which we will define more formally later in the book.

Distributions

The R markdown document for this section is available here[28].

We have explained what we mean by *null* in the context of null hypothesis, but what exactly is a distribution? The simplest way to think of a *distribution* is as a compact description of many numbers. For example, suppose you have measured the heights of all men in a population. Imagine you need to describe these numbers to someone that has no idea what these heights are, such as an alien that has never visited Earth. Suppose all these heights are contained in the following dataset:

```
library(UsingR)
x <- father.son$fheight
```

One approach to summarizing these numbers is to simply list them all out for the alien to see. Here are 10 randomly selected heights of 1,078:

```
round(sample(x,10),1)
```

[28] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

```
##  [1] 67.4 64.9 62.9 69.2 72.3 69.3 65.9 65.2 69.8 69.1
```

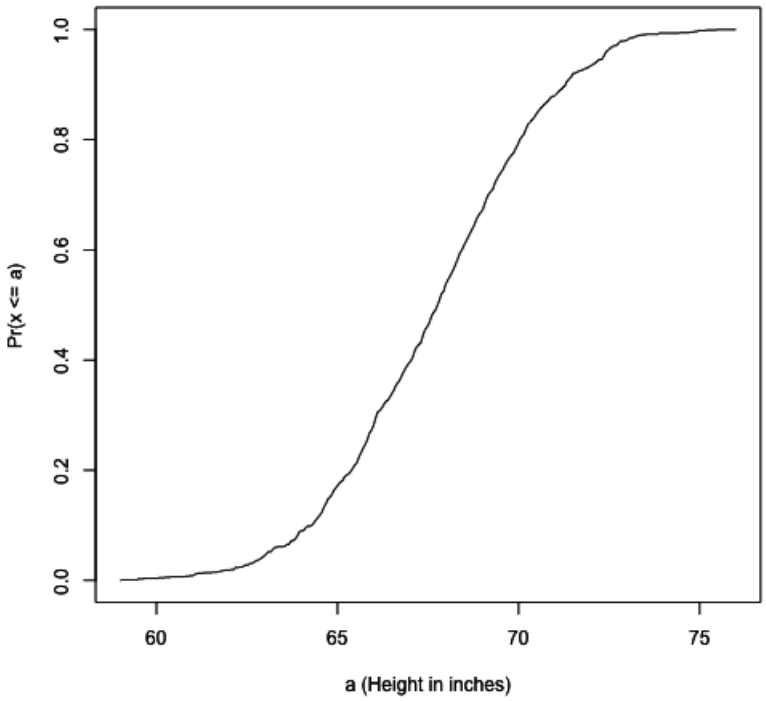
Cumulative Distribution Function

Scanning through these numbers, we start to get a rough idea of what the entire list looks like, but it is certainly inefficient. We can quickly improve on this approach by defining and visualizing a *distribution*. To define a distribution we compute, for all possible values of a, the proportion of numbers in our list that are below a. We use the following notation:

$$F(a) \equiv \Pr(x \leq a)$$

This is called the cumulative distribution function (CDF). When the CDF is derived from data, as opposed to theoretically, we also call it the empirical CDF (ECDF). We can plot $F(a)$ versus a like this:

```
smallest <- floor( min(x) )
largest <- ceiling( max(x) )
values <- seq(smallest, largest,len=300)
heightecdf <- ecdf(x)
plot(values, heightecdf(values), type="l",
     xlab="a (Height in inches)",ylab="Pr(x <= a)")
```



Empirical cummulative distribution function for height.

Histograms

The `ecdf` function is a function that returns a function, which is not typical behavior of R functions. For that reason, we won't discuss it further here. Furthermore, the `ecdf` is actually not as popular as histograms, which give us the same information, but show us the proportion of values in intervals:

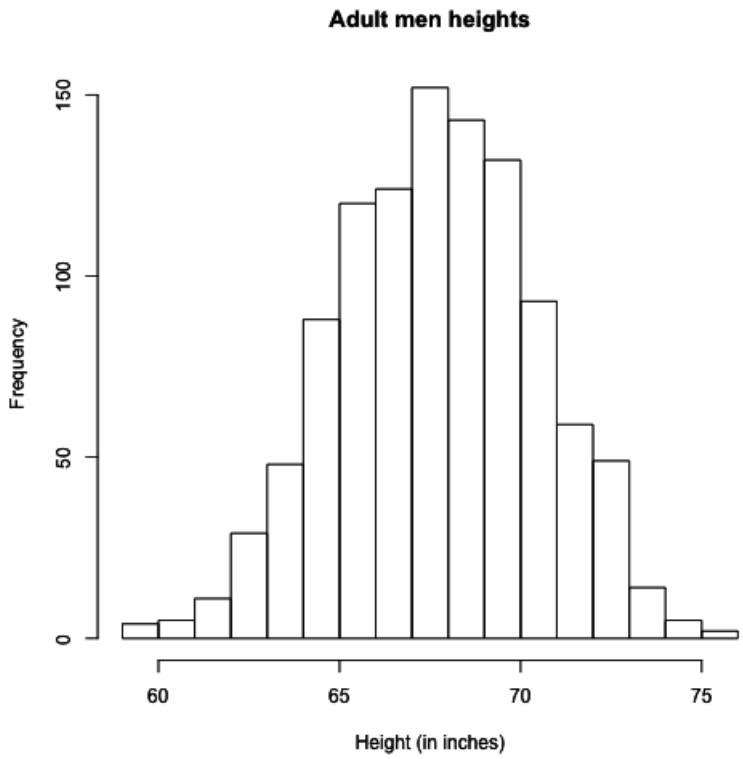
$$\Pr(a \leq x \leq b) = F(b) - F(a)$$

Plotting these heights as bars is what we call a *histogram*. It is a more useful plot because we are usually more interested in intervals, such and such percent are between 70 inches and 71 inches, etc., rather than the percent less than a particular height. It is also easier to distinguish different types (families) of distributions by looking at histograms. Here is a histogram of heights:

```
hist(x)
```

We can specify the bins and add better labels in the following way:

```
bins <- seq(smallest, largest)
hist(x,breaks=bins,xlab="Height (in inches)",main="Adult men heights")
```



Histogram for heights.

Showing this plot to the alien is much more informative than showing numbers. With this simple plot, we can approximate the number of individuals in any given interval. For example, there are about 70 individuals over six feet (72 inches) tall.

Probability Distribution

The R markdown document for this section is available here[29].

Summarizing lists of numbers is one powerful use of distribution. An even more important use is describing the possible outcomes of a random variable. Unlike a fixed list of numbers, we don't actually observe all possible outcomes of random variables, so instead of describing proportions, we describe probabilities. For instance, if we pick a random height from our list, then the probability of it falling between a and b is denoted with:

$$\Pr(a \leq X \leq b) = F(b) - F(a)$$

[29] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

Note that the X is now capitalized to distinguish it as a random variable and that the equation above defines the probability distribution of the random variable. Knowing this distribution is incredibly useful in science. For example, in the case above, if we know the distribution of the difference in mean of mouse weights when the null hypothesis is true, referred to as the *null distribution*, we can compute the probability of observing a value as large as we did, referred to as a *p-value*. In a previous section we ran what is called a *Monte Carlo* simulation (we will provide more details on Monte Carlo simulation in a later section) and we obtained 10,000 outcomes of the random variable under the null hypothesis. Let's repeat the loop above, but this time let's add a point to the figure every time we re-run the experiment. If you run this code, you can see the null distribution forming as the observed values stack on top of each other.

```
n <- 100
library(rafalib)
nullplot(-5,5,1,30, xlab="Observed differences (grams)", ylab="Frequency")
totals <- vector("numeric",11)
for (i in 1:n) {
  control <- sample(population,12)
  treatment <- sample(population,12)
  nulldiff <- mean(treatment) - mean(control)
  j <- pmax(pmin(round(nulldiff)+6,11),1)
  totals[j] <- totals[j]+1
  text(j-6,totals[j],pch=15,round(nulldiff,1))
  ##if(i < 15) Sys.sleep(1) ##You can add this line to see values appear slowly
  }
```

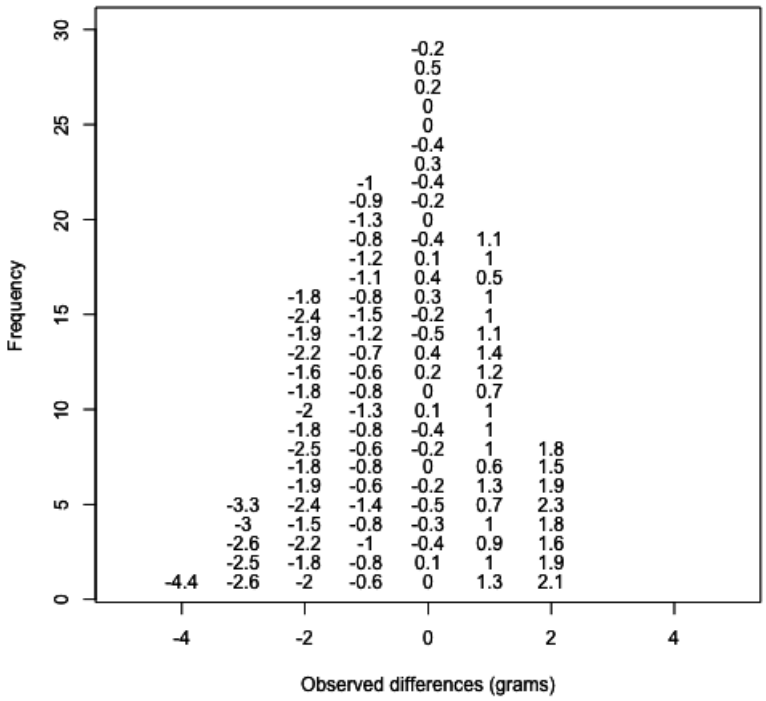
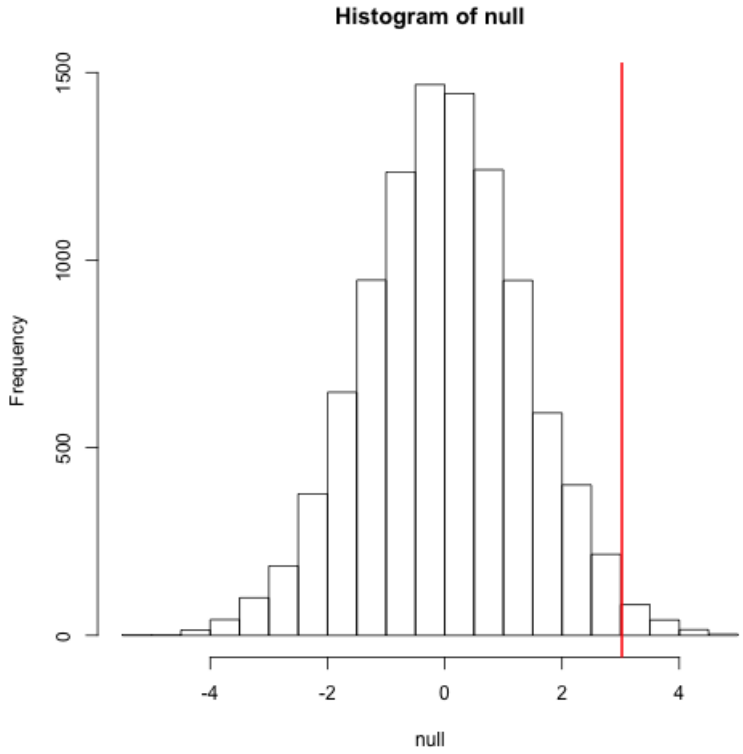


Illustration of the null distribution.

The figure above amounts to a histogram. From a histogram of the `null` vector we calculated earlier, we can see that values as large as `obsdiff` are relatively rare:

```
hist(null, freq=TRUE)
abline(v=obsdiff, col="red", lwd=2)
```



Null distribution with observed difference marked with vertical red line.

An important point to keep in mind here is that while we defined $\Pr(a)$ by counting cases, we will learn that, in some circumstances, mathematics gives us formulas for $\Pr(a)$ that save us the trouble of computing them as we did here. One example of this powerful approach uses the normal distribution approximation.

Normal Distribution

The R markdown document for this section is available here[30].

The probability distribution we see above approximates one that is very common in nature: the bell curve, also known as the normal distribution or Gaussian distribution. When the histogram of a list of numbers approximates the normal distribution, we can use a convenient mathematical formula to approximate the proportion of values or outcomes in any given interval:

$$\Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) dx$$

[30] https://github.com/genomicsclass/labs/tree/master/inference/random_variables.Rmd

While the formula may look intimidating, don't worry, you will never actually have to type it out, as it is stored in a more convenient form (as `pnorm` in R which sets a to $-\infty$, and takes b as an argument).

Here μ and σ are referred to as the mean and the standard deviation of the population (we explain these in more detail in another section). If this *normal approximation* holds for our list, then the population mean and variance of our list can be used in the formula above. An example of this would be when we noted above that only 1.5% of values on the null distribution were above `obsdiff`. We can compute the proportion of values below a value `x` with `pnorm(x,mu,sigma)` without knowing all the values. The normal approximation works very well here:

```
1 - pnorm(obsdiff,mean(null),sd(null))
```

```
## [1] 0.01391929
```

Later, we will learn that there is a mathematical explanation for this. A very useful characteristic of this approximation is that one only needs to know μ and σ to describe the entire distribution. From this, we can compute the proportion of values in any interval.

Summary

So computing a p-value for the difference in diet for the mice was pretty easy, right? But why are we not done? To make the calculation, we did the equivalent of buying all the mice available from The Jackson Laboratory and performing our experiment repeatedly to define the null distribution. Yet this is not something we can do in practice. Statistical Inference is the mathematical theory that permits you to approximate this with only the data from your sample, i.e. the original 24 mice. We will focus on this in the following sections.

Setting the random seed

Before we continue, we briefly explain the following important line of code:

```
set.seed(1)
```

Throughout this book, we use random number generators. This implies that many of the results presented can actually change by chance, including the correct answer to problems. One way to ensure that results do not change is by setting R's random number generation seed. For more on the topic please read the help file:

```
?set.seed
```

Exercises

For these exercises, we will be using the following dataset:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/femaleControlsPopulation.csv"
filename <- basename(url)
download(url, destfile=filename)
x <- unlist( read.csv(filename) )
```

Here `x` represents the weights for the entire population.

1. What is the average of these weights?
2. After setting the seed at 1, `set.seed(1)` take a random sample of size 5. What is the absolute value (use `abs`) of the difference between the average of the sample and the average of all the values?
3. After setting the seed at 5, `set.seed(5)` take a random sample of size 5. What is the absolute value of the difference between the average of the sample and the average of all the values?
4. Why are the answers from 2 and 3 different?
    - A) Because we made a coding mistake.
    - B) Because the average of the `x` is random.
    - C) Because the average of the samples is a random variable.
    - D) All of the above.
5. Set the seed at 1, then using a for-loop take a random sample of 5 mice 1,000 times. Save these averages. What percent of these 1,000 averages are more than 1 ounce away from the average of `x` ?
6. We are now going to increase the number of times we redo the sample from 1,000 to 10,000. Set the seed at 1, then using a for-loop take a random sample of 5 mice 10,000 times. Save these averages. What percent of these 10,000 averages are more than 1 ounce away from the average of `x` ?
7. Note that the answers to 4 and 5 barely changed. This is expected. The way we think about the random value distributions is as the distribution of the list of values obtained if we repeated the experiment an infinite number of times. On a computer, we can't perform an infinite number of iterations so instead, for our examples, we consider 1,000 to be large enough, thus 10,000 is as well. Now if instead we change the sample size, then we change the random variable and thus its distribution.

   Set the seed at 1, then using a for-loop take a random sample of 50 mice 1,000 times. Save these averages. What percent of these 1,000 averages are more than 1 ounce away from the average of `x` ?

8. Use a histogram to "look" at the distribution of averages we get with a sample size of 5 and a sample size of 50. How would you say they differ?
    - A) They are actually the same.
    - B) They both look roughly normal, but with a sample size of 50 the spread is smaller.
    - C) They both look roughly normal, but with a sample size of 50 the spread is larger.
    - D) The second distribution does not look normal at all.
9. For the last set of averages, the ones obtained from a sample size of 50, what percent are between 23 and 25?
10. Now ask the same question of a normal distribution with average 23.9 and standard deviation 0.43.

The answer to 9 and 10 were very similar. This is because we can approximate the distribution of the sample average with a normal distribution. We will learn more about the reason for this next.

Populations, Samples and Estimates

The R markdown document for this section is available here[31].

Now that we have introduced the idea of a random variable, a null distribution, and a p-value, we are ready to describe the mathematical theory that permits us to compute p-values in practice. We will also learn about confidence intervals and power calculations.

Population parameters

A first step in statistical inference is to understand what population you are interested in. In the mouse weight example, we have two populations: female mice on control diets and female mice on high fat diets, with weight being the outcome of interest. We consider this population to be fixed, and the randomness comes from the sampling. One reason we have been using this dataset as an example is because we happen to have the weights of all the mice of this type. Here we download and read in this dataset:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/mice_pheno.csv"
filename <- "mice_pheno.csv"
download(url,destfile=filename)
dat <- read.csv(filename)
```

We can then access the population values and determine, for example, how many we have. Here we compute the size of the control population:

[31] https://github.com/genomicsclass/labs/tree/master/inference/populations_and_samples.Rmd

```
library(dplyr)
controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
  select(Bodyweight) %>% unlist
length(controlPopulation)
```

```
## [1] 225
```

We usually denote these values as $x_1, \ldots, x_m$. In this case, m is the number computed above. We can do the same for the high fat diet population:

```
hfPopulation <- filter(dat,Sex == "F" & Diet == "hf") %>%
  select(Bodyweight) %>% unlist
length(hfPopulation)
```

```
## [1] 200
```

and denote with $y_1, \ldots, y_n$.

We can then define summaries of interest for these populations, such as the mean and variance.

The mean:

$$\mu_X = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ and } \mu_Y = \frac{1}{n}\sum_{i=1}^{n} y_i$$

The variance:

$$\sigma_X^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_X)^2 \text{ and } \sigma_Y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_Y)^2$$

with the standard deviation being the square root of the variance. We refer to such quantities that can be obtained from the population as *population parameters*. The question we started out asking can now be written mathematically: is $\mu_Y - \mu_X = 0$?

Although in our illustration we have all the values and can check if this is true, in practice we do not. For example, in practice it would be prohibitively expensive to buy all the mice in a population. Here we learn how taking a *sample* permits us to answer our questions. This is the essence of statistical inference.

Sample estimates

In the previous chapter, we obtained samples of 12 mice from each population. We represent data from samples with capital letters to indicate that they are random. This is common practice in statistics, although it is not always followed. So the samples are $X_1, \ldots, X_M$ and $Y_1, \ldots, Y_N$ and, in this case, $N = M = 12$. In contrast and as we saw above, when we list out the values of the population, which are set and not random, we use lower-case letters.

Since we want to know if $\mu_Y - \mu_X$ is 0, we consider the sample version: $\bar{Y} - \bar{X}$ with:

$$\bar{X} = \frac{1}{M}\sum_{i=1}^{M} X_i \text{ and } \bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i.$$

Note that this difference of averages is also a random variable. Previously, we learned about the behavior of random variables with an exercise that involved repeatedly sampling from the original distribution. Of course, this is not an exercise that we can execute in practice. In this particular case it would involve buying 24 mice over and over again. Here we described the mathematical theory that mathematically relates $\bar{X}$ to μ_X and $\bar{Y}$ to μ_Y, that will in turn help us understand the relationship between $\bar{Y} - \bar{X}$ and $\mu_Y - \mu_X$. Specifically, we will describe how the Central Limit Theorem permits us to use an approximation to answer this question, as well as motivate the widely used t-distribution.

Exercises

For these exercises, we will be using the following dataset:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/mice_pheno.csv"
filename <- basename(url)
download(url, destfile=filename)
dat <- read.csv(filename)
```

We will remove the lines that contain missing values:

```
dat <- na.omit( dat )
```

1. Use `dplyr` to create a vector `x` with the body weight of all males on the control (`chow`) diet. What is this population's average?
2. Now use the `rafalib` package and use the `popsd` function to compute the population standard deviation.
3. Set the seed at 1. Take a random sample X of size 25 from `x`. What is the sample average?

4. Use `dplyr` to create a vector `y` with the body weight of all males on the high fat (`hf`) diet. What is this population's average?
5. Now use the `rafalib` package and use the `popsd` function to compute the population standard deviation.
6. Set the seed at 1. Take a random sample Y of size 25 from `y`. What is the sample average?
7. What is the difference in absolute value between $\bar{y} - \bar{x}$ and $\bar{X} - \bar{Y}$?
8. Repeat the above for females. Make sure to set the seed to 1 before each `sample` call. What is the difference in absolute value between $\bar{y} - \bar{x}$ and $\bar{X} - \bar{Y}$?
9. For the females, our sample estimates were closer to the population difference than with males. What is a possible explanation for this?
    - A) The population variance of the females is smaller than that of the males; thus, the sample variable has less variability.
    - B) Statistical estimates are more precise for females.
    - C) The sample size was larger for females.
    - D) The sample size was smaller for females.

Central Limit Theorem and t-distribution

The R markdown document for this section is available here[32].

Below we will discuss the Central Limit Theorem (CLT) and the t-distribution, both of which help us make important calculations related to probabilities. Both are frequently used in science to test statistical hypotheses. To use these, we have to make different assumptions from those for the CLT and the t-distribution. However, if the assumptions are true, then we are able to calculate the exact probabilities of events through the use of mathematical formula.

Central Limit Theorem

The CLT is one of the most frequently used mathematical results in science. It tells us that when the sample size is large, the average $\bar{Y}$ of a random sample follows a normal distribution centered at the population average μ_Y and with standard deviation equal to the population standard deviation σ_Y, divided by the square root of the sample size N. We refer to the standard deviation of the distribution of a random variable as the random variable's *standard error*.

Please note that if we subtract a constant from a random variable, the mean of the new random variable shifts by that constant. Mathematically, if X is a random variable with mean μ and a is a constant, the mean of $X - a$ is $\mu - a$. A similarly intuitive result holds for multiplication and the standard deviation (SD). If X is a random variable with mean μ and SD σ, and a is a constant, then the mean and SD of aX are $a\mu$ and $\mid a \mid \sigma$ respectively. To see how intuitive this is, imagine that

[32] https://github.com/genomicsclass/labs/tree/master/inference/clt_and_t-distribution.Rmd

we subtract 10 grams from each of the mice weights. The average weight should also drop by that much. Similarly, if we change the units from grams to milligrams by multiplying by 1000, then the spread of the numbers becomes larger.

This implies that if we take many samples of size N, then the quantity:

$$\frac{\bar{Y} - \mu}{\sigma_Y/\sqrt{N}}$$

is approximated with a normal distribution centered at 0 and with standard deviation 1.

Now we are interested in the difference between two sample averages. Here again a mathematical result helps. If we have two random variables X and Y with means μ_X and μ_Y and variance σ_X and σ_Y respectively, then we have the following result: the mean of the sum $Y + X$ is the sum of the means $\mu_Y + \mu_X$. Using one of the facts we mentioned earlier, this implies that the mean of $Y - X = Y + aX$ with $a = -1$, which implies that the mean of $Y - X$ is $\mu_Y - \mu_X$. This is intuitive. However, the next result is perhaps not as intuitive. If X and Y are independent of each other, as they are in our mouse example, then the variance (SD squared) of $Y + X$ is the sum of the variances $\sigma_Y^2 + \sigma_X^2$. This implies that variance of the difference $Y - X$ is the variance of $Y + aX$ with $a = -1$ which is $\sigma_Y^2 + a^2\sigma_X^2 = \sigma_Y^2 + \sigma_X^2$. So the variance of the difference is also the sum of the variances. If this seems like a counterintuitive result, remember that if X and Y are independent of each other, the sign does not really matter. It can be considered random: if X is normal with certain variance, for example, so is $-X$. Finally, another useful result is that the sum of normal variables is again normal.

All this math is very helpful for the purposes of our study because we have two sample averages and are interested in the difference. Because both are normal, the difference is normal as well, and the variance (the standard deviation squared) is the sum of the two variances. Under the null hypothesis that there is no difference between the population averages, the difference between the sample averages $\bar{Y} - \bar{X}$, with $\bar{X}$ and $\bar{Y}$ the sample average for the two diets respectively, is approximated by a normal distribution centered at 0 (there is no difference) and with standard deviation $\sqrt{\sigma_X^2 + \sigma_Y^2}/\sqrt{N}$.

This suggests that this ratio:

$$\frac{\bar{Y} - \bar{X}}{\sqrt{\frac{\sigma_X^2}{M} + \frac{\sigma_Y^2}{N}}}$$

is approximated by a normal distribution centered at 0 and standard deviation 1. Using this approximation makes computing p-values simple because we know the proportion of the distribution under any value. For example, only 5% of these values are larger than 2 (in absolute value):

```
pnorm(-2) + (1 - pnorm(2))
```

```
## [1] 0.04550026
```

We don't need to buy more mice, 12 and 12 suffice.

However, we can't claim victory just yet because we don't know the population standard deviations: σ_X and σ_Y. These are unknown population parameters, but we can get around this by using the sample standard deviations, call them s_X and s_Y. These are defined as:

$$s_X^2 = \frac{1}{M-1}\sum_{i=1}^{M}(X_i - \bar{X})^2 \text{ and } s_Y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})^2$$

Note that we are dividing by $M-1$ and $N-1$, instead of by M and N. There is a theoretical reason for doing this which we do not explain here. But to get an intuition, think of the case when you just have 2 numbers. The average distance to the mean is basically 1/2 the difference between the two numbers. So you really just have information from one number. This is somewhat of a minor point. The main point is that s_X and s_Y serve as estimates of σ_X and σ_Y

So we can redefine our ratio as

$$\sqrt{N}\frac{\bar{Y} - \bar{X}}{\sqrt{s_X^2 + s_Y^2}}$$

if $M = N$ or in general,

$$\frac{\bar{Y} - \bar{X}}{\sqrt{\frac{s_X^2}{M} + \frac{s_Y^2}{N}}}$$

The CLT tells us that when M and N are large, this random variable is normally distributed with mean 0 and SD 1. Thus we can compute p-values using the function `pnorm`.

The t-distribution

The CLT relies on large samples, what we refer to as *asymptotic results*. When the CLT does not apply, there is another option that does not rely on asymptotic results. When the original population from which a random variable, say Y, is sampled is normally distributed with mean 0, then we can calculate the distribution of:

$$\sqrt{N}\frac{\bar{Y}}{s_Y}$$

This is the ratio of two random variables so it is not necessarily normal. The fact that the denominator can be small by chance increases the probability of observing large values. William Sealy Gosset[33],

[33] http://en.wikipedia.org/wiki/William_Sealy_Gosset

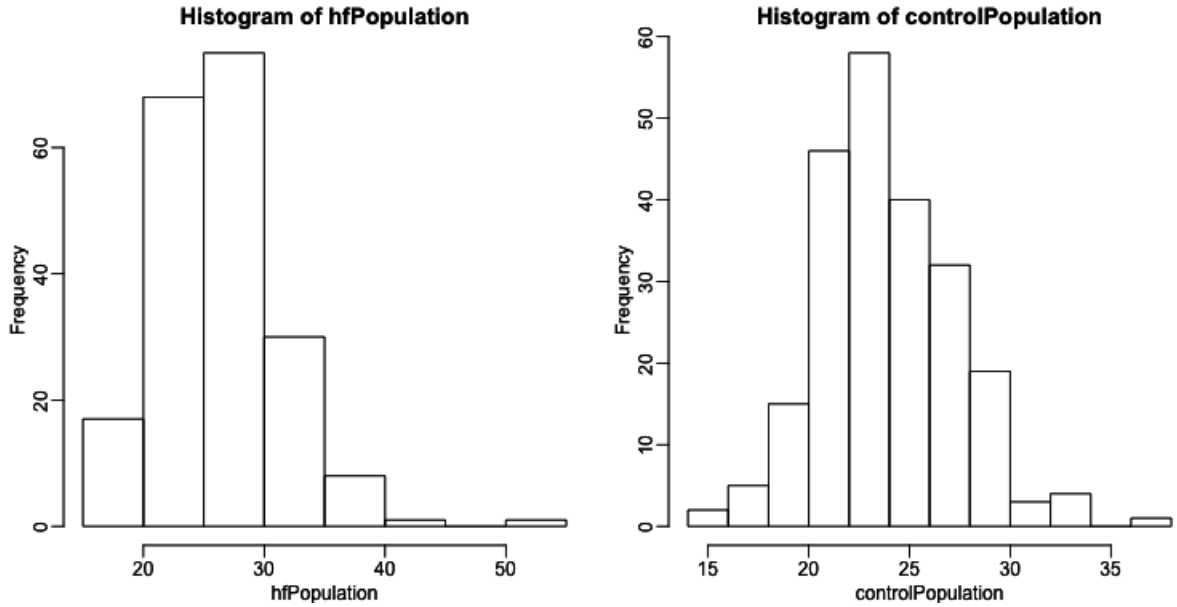
an employee of the Guinness brewing company, deciphered the distribution of this random variable and published a paper under the pseudonym "Student". The distribution is therefore called Student's t-distribution. Later we will learn more about how this result is used.

Here we will use the mice phenotype data as an example. We start by creating two vectors, one for the control population and one for the high-fat diet population:

```
library(dplyr)
dat <- read.csv("mice_pheno.csv") #We downloaded this file in a previous section
controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
  select(Bodyweight) %>% unlist
hfPopulation <- filter(dat,Sex == "F" & Diet == "hf") %>%
  select(Bodyweight) %>% unlist
```

It is important to keep in mind that what we are assuming to be normal here is the distribution of $y_1, y_2, \ldots, y_n$, not the random variable $\bar{Y}$. Although we can't do this in practice, in this illustrative example, we get to see this distribution for both controls and high fat diet mice:

```
library(rafalib)
mypar(1,2)
hist(hfPopulation)
hist(controlPopulation)
```

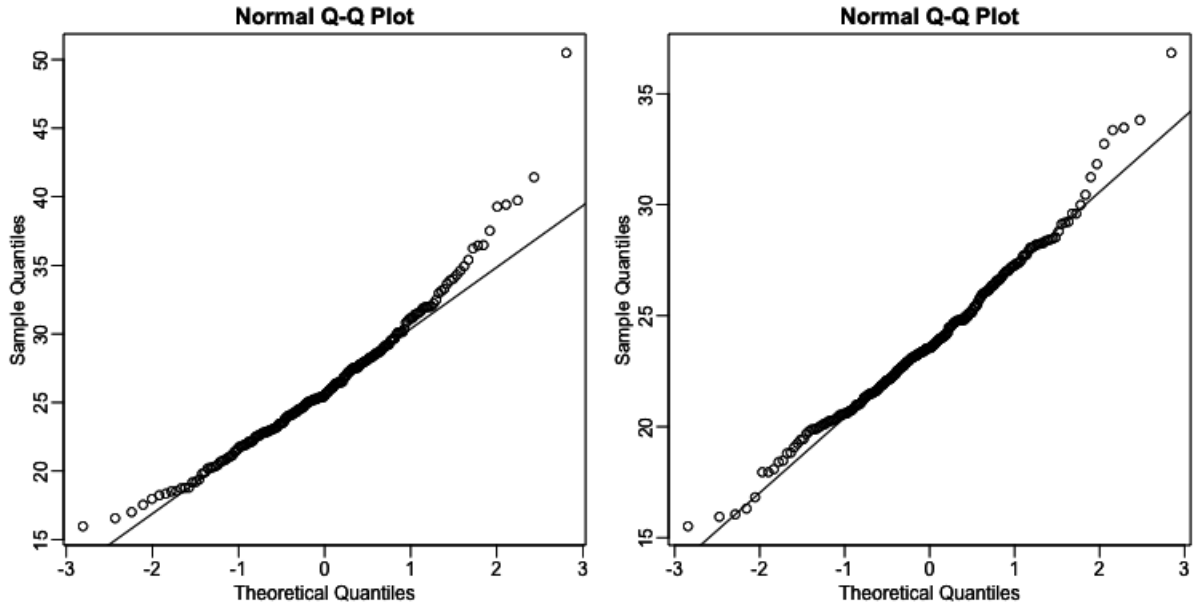


Histograms of all weights for both populations.

We can use *qq-plots* to confirm that the distributions are relatively close to being normally distributed. We will explore these plots in more depth in a later section, but the important thing

to know is that it compares data (on the y-axis) against a theoretical distribution (on the x-axis). If the points fall on the identity line, then the data is close to the theoretical distribution.

```
mypar(1,2)
qqnorm(hfPopulation)
qqline(hfPopulation)
qqnorm(controlPopulation)
qqline(controlPopulation)
```



Quantile-quantile plots of all weights for both populations.

The larger the sample, the more forgiving the result is to the weakness of this approximation. In the next section, we will see that for this particular dataset the t-distribution works well even for sample sizes as small as 3.

Exercises

For these exercises, we will be using the following dataset:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/mice_pheno.csv"
filename <- basename(url)
download(url, destfile=filename)
dat <- na.omit( read.csv(filename) )
```

1. If a list of numbers has a distribution that is well approximated by the normal distribution, what proportion of these numbers are within one standard deviation away from the list's average?
2. What proportion of these numbers are within two standard deviations away from the list's average?
3. What proportion of these numbers are within three standard deviations away from the list's average?
4. Define `y` to be the weights of males on the control diet. What proportion of the mice are within one standard deviation away from the average weight (remember to use `popsd` for the population `sd`)?
5. What proportion of these numbers are within two standard deviations away from the list's average?
6. What proportion of these numbers are within three standard deviations away from the list's average?
7. Note that the numbers for the normal distribution and our weights are relatively close. Also, notice that we are indirectly comparing quantiles of the normal distribution to quantiles of the mouse weight distribution. We can actually compare all quantiles using a qqplot. Which of the following best describes the qq-plot comparing mouse weights to the normal distribution?
    - A) The points on the qq-plot fall exactly on the identity line.
    - B) The average of the mouse weights is not `0` and thus it can't follow a normal distribution.
    - C) The mouse weights are well approximated by the normal distribution, although the larger values (right tail) are larger than predicted by the normal. This is consistent with the differences seen between question 3 and 6.
    - D) These are not random variables and thus they can't follow a normal distribution.
8. Create the above qq-plot for the four populations: male/females on each of the two diets. What is the most likely explanation for the mouse weights being well approximated? What is the best explanation for all these being well approximated by the normal distribution?
    - A) The CLT tells us that sample averages are approximately normal.
    - B) This just happens to be how nature behaves. Perhaps the result of many biological factors averaging out.
    - C) Everything measured in nature follows a normal distribution.
    - D) Measurement error is normally distributed.
9. Here we are going to use the function `replicate` to learn about the distribution of random variables. All the above exercises relate to the normal distribution as an approximation of the distribution of a fixed list of numbers or a population. We have not yet discussed probability in these exercises. If the distribution of a list of numbers is approximately normal, then if we pick a number at random from this distribution, it will follow a normal distribution. However, it is important to remember that stating that some quantity has a distribution does not necessarily imply this quantity is random. Also, keep in mind that this is not related to the central limit theorem. The central limit applies to averages of random variables. Let's explore this concept.

We will now take a sample of size 25 from the population of males on the chow diet. The average of this sample is our random variable. We will use the `replicate` to observe 10,000 realizations of this random variable. Set the seed at 1, generate these 10,000 averages. Make a histogram and qq-plot of these 10,000 numbers against the normal distribution.

We can see that, as predicted by the CLT, the distribution of the random variable is very well approximated by the normal distribution.

```
y <- filter(dat, Sex=="M" & Diet=="chow") %>% select(Bodyweight) %>% unlist
avgs <- replicate(10000, mean( sample(y, 25)))
mypar(1,2)
hist(avgs)
qqnorm(avgs)qqline(avgs)
```

What is the average of the distribution of the sample average?

10. What is the standard deviation of the distribution of sample averages?
11. According to the CLT, the answer to exercise 9 should be the same as `mean(y)`. You should be able to confirm that these two numbers are very close. Which of the following does the CLT tell us should be close to your answer to exercise 10?
    - A) `popsd(y)`
    - B) `popsd(avgs)/sqrt(25)`
    - C) `sqrt(25) / popsd(y)`
    - D) `popsd(y)/sqrt(25)`
12. In practice we do not know σ (`popsd(y)`) which is why we can't use the CLT directly. This is because we see a sample and not the entire distribution. We also can't use `popsd(avgs)` because to construct averages, we have to take 10,000 samples and this is never practical. We usually just get one sample. Instead we have to estimate `popsd(y)`. As described, what we use is the sample standard deviation. Set the seed at 1, using the `replicate` function, create 10,000 samples of 25 and now, instead of the sample average, keep the standard deviation. Look at the distribution of the sample standard deviations. It is a random variable. The real population SD is about 4.5. What proportion of the sample SDs are below 3.5?
13. What the answer to question 12 reveals is that the denominator of the t-test is a random variable. By decreasing the sample size, you can see how this variability can increase. It therefore adds variability. The smaller the sample size, the more variability is added. The normal distribution stops providing a useful approximation. When the distribution of the population values is approximately normal, as it is for the weights, the t-distribution provides a better approximation. We will see this later on. Here we will look at the difference between the t-distribution and normal. Use the function `qt` and `qnorm` to get the quantiles of `x=seq(0.0001,0.9999,len=300)`. Do this for degrees of freedom 3, 10, 30, and 100. Which of the following is true?
    - A) The t-distribution and normal distribution are always the same.
    - B) The t-distribution has a higher average than the normal distribution.
    - C) The t-distribution has larger tails up until 30 degrees of freedom, at which point it is practically the same as the normal distribution.

- D) The variance of the t-distribution grows as the degrees of freedom grow.

Central Limit Theorem in Practice

The R markdown document for this section is available here[34].

Let's use our data to see how well the central limit theorem approximates sample averages from our data. We will leverage our entire population dataset to compare the results we obtain by actually sampling from the distribution to what the CLT predicts.

```
dat <- read.csv("mice_pheno.csv") #file was previously downloaded
head(dat)
```

```
##   Sex Diet Bodyweight
## 1   F   hf      31.94
## 2   F   hf      32.48
## 3   F   hf      22.82
## 4   F   hf      19.92
## 5   F   hf      32.22
## 6   F   hf      27.50
```

Start by selecting only female mice since males and females have different weights. We will select three mice from each population.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

[34] https://github.com/genomicsclass/labs/tree/master/inference/clt_in_practice.Rmd

```
controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
  select(Bodyweight) %>% unlist
hfPopulation <- filter(dat,Sex == "F" & Diet == "hf") %>%
  select(Bodyweight) %>% unlist
```

We can compute the population parameters of interest using the mean function.

```
mu_hf <- mean(hfPopulation)
mu_control <- mean(controlPopulation)
print(mu_hf - mu_control)
```

```
## [1] 2.375517
```

Compute the population standard deviations as well. We do not use the R function `sd` because this would compute the estimates that divide by the sample size - 1 and we want the population estimates.

We can see that with R code:

```
x <- controlPopulation
N <- length(x)
populationvar <- mean((x-mean(x))^2)
identical(var(x), populationvar)
```

```
## [1] FALSE
```

```
identical(var(x)*(N-1)/N, populationvar)
```

```
## [1] TRUE
```

So to be mathematically correct, we do not use `sd` or `var`. Instead, we use the `popvar` and `popsd` function in `rafalib`:

```
library(rafalib)
sd_hf <- popsd(hfPopulation)
sd_control <- popsd(controlPopulation)
```

Remember that in practice we do not get to compute these population parameters. These are values we never see. In general, we want to estimate them from samples.

```
N <- 12
hf <- sample(hfPopulation, 12)
control <- sample(controlPopulation, 12)
```

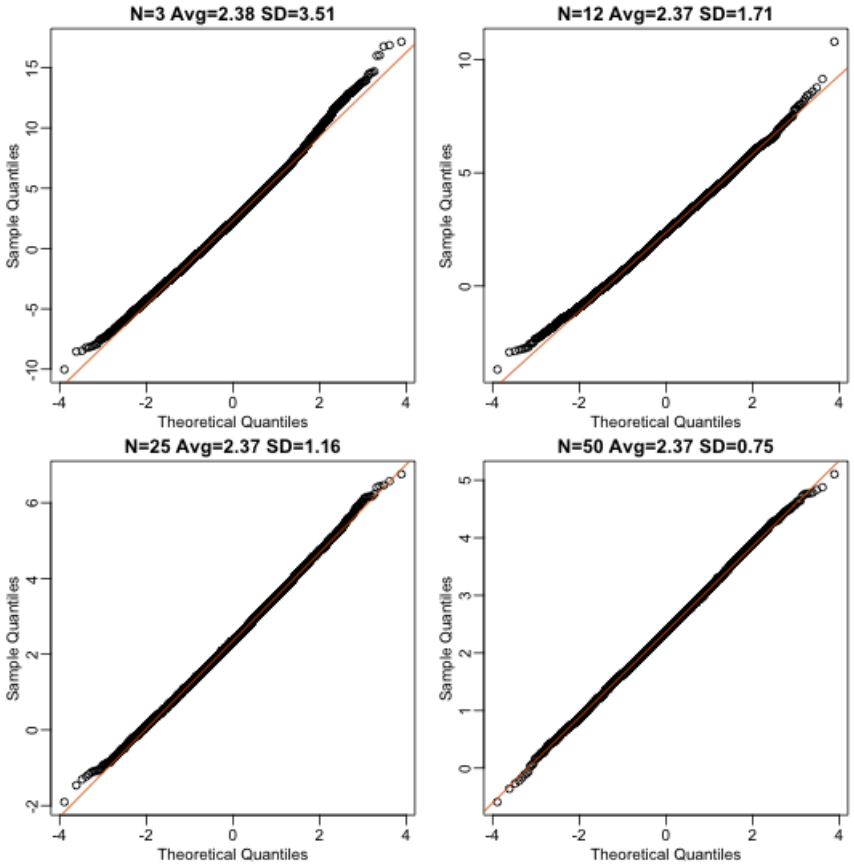
As we described, the CLT tells us that for large N, each of these is approximately normal with average population mean and standard error population variance divided by N. We mentioned that a rule of thumb is that N should be 30 or more. However, that is just a rule of thumb since the preciseness of the approximation depends on the population distribution. Here we can actually check the approximation and we do that for various values of N.

Now we use `sapply` and `replicate` instead of `for` loops, which makes for cleaner code (we do not have to pre-allocate a vector, R takes care of this for us):

```
Ns <- c(3,12,25,50)
B <- 10000 #number of simulations
res <-  sapply(Ns,function(n) {
  replicate(B,mean(sample(hfPopulation,n))-mean(sample(controlPopulation,n)))
})
```

Now we can use qq-plots to see how well CLT approximations works for these. If in fact the normal distribution is a good approximation, the points should fall on a straight line when compared to normal quantiles. The more it deviates, the worse the approximation. In the title, we also show the average and SD of the observed distribution, which demonstrates how the SD decreases with $\sqrt{N}$ as predicted.

```
mypar(2,2)
for (i in seq(along=Ns)) {
  titleavg <- signif(mean(res[,i]),3)
  titlesd <- signif(popsd(res[,i]),3)
  title <- paste0("N=",Ns[i]," Avg=",titleavg," SD=",titlesd)
  qqnorm(res[,i],main=title)
  qqline(res[,i],col=2)
}
```

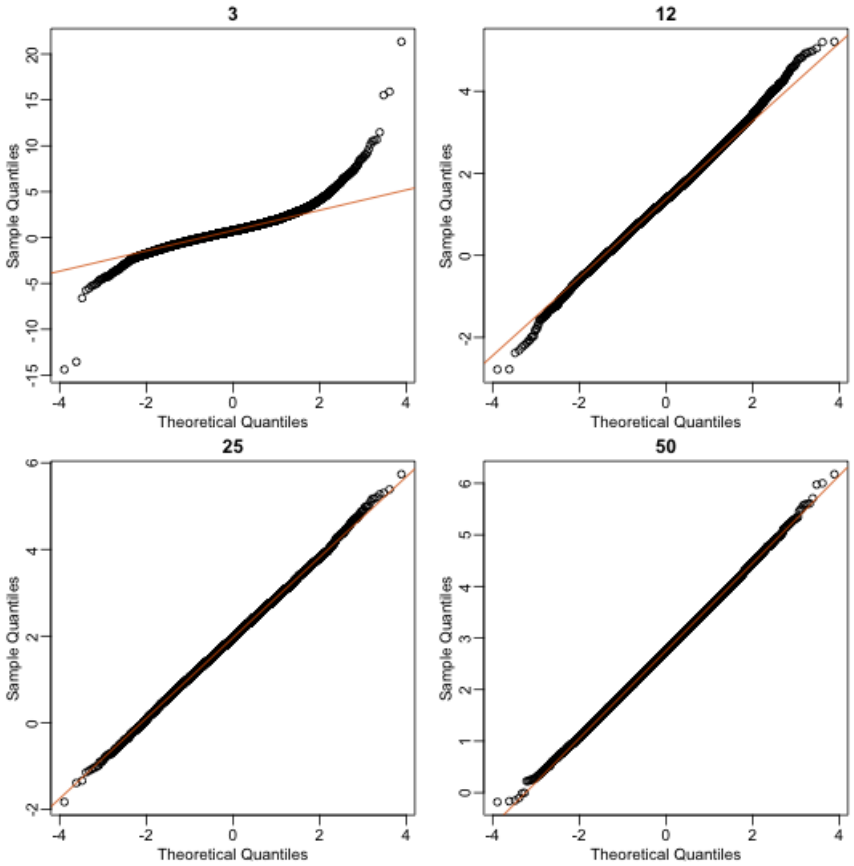


Quantile versus quantile plot of simulated differences versus theoretical normal distribution for four different sample sizes.

Here we see a pretty good fit even for 3. Why is this? Because the population itself is relatively close to normally distributed, the averages are close to normal as well (the sum of normals is also a normal). In practice, we actually calculate a ratio: we divide by the estimated standard deviation. Here is where the sample size starts to matter more.

```
Ns <- c(3,12,25,50)
B <- 10000 #number of simulations
##function to compute a t-stat
computetstat <- function(n) {
  y <- sample(hfPopulation,n)
  x <- sample(controlPopulation,n)
  (mean(y)-mean(x))/sqrt(var(y)/n+var(x)/n)
}
res <-  sapply(Ns,function(n) {
  replicate(B,computetstat(n))
})
mypar(2,2)
```

```
for (i in seq(along=Ns)) {
  qqnorm(res[,i],main=Ns[i])
  qqline(res[,i],col=2)
}
```



Quantile versus quantile plot of simulated ratios versus theoretical normal distribution for four different sample sizes.

So we see that for $N = 3$, the CLT does not provide a usable approximation. For $N = 12$, there is a slight deviation at the higher values, although the approximation appears useful. For 25 and 50, the approximation is spot on.

This simulation only proves that $N = 12$ is large enough in this case, not in general. As mentioned above, we will not be able to perform this simulation in most situations. We only use the simulation to illustrate the concepts behind the CLT and its limitations. In future sections, we will describe the approaches we actually use in practice.

Exercises

Exercises 3-13 use the mouse data set we have previously downloaded:

```
library(downloader)
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/femaleMiceWeights.csv"
filename <- "femaleMiceWeights.csv"
if(!file.exists("femaleMiceWeights.csv")) download(url,destfile=filename)
dat <- read.csv(filenames)
```

1. The CLT is a result from probability theory. Much of probability theory was originally inspired by gambling. This theory is still used in practice by casinos. For example, they can estimate how many people need to play slots for there to be a 99.9999% probability of earning enough money to cover expenses. Let's try a simple example related to gambling.

   Suppose we are interested in the proportion of times we see a 6 when rolling `n=100` die. This is a random variable which we can simulate with `x=sample(1:6, n, replace=TRUE)` and the proportion we are interested in can be expressed as an average: `mean(x==6)`. Because the die rolls are independent, the CLT applies.

   We want to roll `n` dice 10,000 times and keep these proportions. This random variable (proportion of 6s) has mean `p=1/6` and variance `p*(1-p)/n`. So according to CLT `z = (mean(x==6) - p) / sqrt(p*(1-p)/n)` should be normal with mean 0 and SD 1. Set the seed to 1, then use `replicate` to perform the simulation, and report what proportion of times `z` was larger than 2 in absolute value (CLT says it should be about 0.05).
2. For the last simulation you can make a qqplot to confirm the normal approximation. Now, the CLT is an *asympototic* result, meaning it is closer and closer to being a perfect approximation as the sample size increases. In practice, however, we need to decide if it is appropriate for actual sample sizes. Is 10 enough? 15? 30?

   In the example used in exercise 1, the original data is binary (either 6 or not). In this case, the success probability also affects the appropriateness of the CLT. With very low probabilities, we need larger sample sizes for the CLT to "kick in".

   Run the simulation from exercise 1, but for different values of `p` and `n`. For which of the following is the normal approximation best?
   - A) `p=0.5` and `n=5`
   - B) `p=0.5` and `n=30`
   - C) `p=0.01` and `n=30`
   - D) `p=0.01` and `n=100`
3. As we have already seen, the CLT also applies to averages of quantitative data. A major difference with binary data, for which we know the variance is $p(1-p)$, is that with quantitative data we need to estimate the population standard deviation.

In several previous exercises we have illustrated statistical concepts with the unrealistic situation of having access to the entire population. In practice, we do *not* have access to entire populations. Instead, we obtain one random sample and need to reach conclusions analyzing that data. `dat` is an example of a typical simple dataset representing just one sample. We have 12 measurements for each of two populations:

```
X <- filter(dat, Diet=="chow") %>% select(Bodyweight) %>% unlist
Y <- filter(dat, Diet=="hf") %>% select(Bodyweight) %>% unlist
```

We think of X as a random sample from the population of all mice in the control diet and Y as a random sample from the population of all mice in the high fat diet.

Define the parameter μ_x as the average of the control population. We estimate this parameter with the sample average $\bar{X}$. What is the sample average?

4. We don't know μ_X , but want to use $\bar{X}$ to understand μ_X. Which of the following uses CLT to understand how well $\bar{X}$ approximates μ_X ?
   - A) $\bar{X}$ follows a normal distribution with mean 0 and standard deviation 1.
   - B) μ_X follows a normal distribution with mean $\bar{X}$ and standard deviation $\frac{\sigma_x}{\sqrt{12}}$ where σ_x is the population standard deviation.
   - C) $\bar{X}$ follows a normal distribution with mean μ_X and standard deviation σ_x where σ_x is the population standard deviation.
   - D) $\bar{X}$ follows a normal distribution with mean μ_X and standard deviation $\frac{\sigma_x}{\sqrt{12}}$ where σ_x is the population standard deviation.
5. The result above tells us the distribution of the following random variable: $Z = \sqrt{12}\frac{\bar{X}-\mu_X}{\sigma_X}$. What does the CLT tell us is the mean of Z (you don't need code)?
6. The result of 4 and 5 tell us that we know the distribution of the difference between our estimate and what we want to estimate, but don't know. However, the equation involves the population standard deviation σ_X, which we don't know. Given what we discussed, what is your estimate of σ_x?
7. Use the CLT to approximate the probability that our estimate $\bar{X}$ is off by more than 5.21 ounces from μ_X.
8. Now we introduce the concept of a null hypothesis. We don't know μ_x nor μ_y. We want to quantify what the data say about the possibility that the diet has no effect: $\mu_x = \mu_y$. If we use CLT, then we approximate the distribution of $\bar{X}$ as normal with mean μ_X and standard deviation σ_X and the distribution of $\bar{Y}$ as normal with mean μ_y and standard deviation σ_y. This implies that the difference $\bar{Y} - \bar{X}$ has mean 0. We described that the standard deviation of this statistic (the standard error) is $\mathrm{SE}(\bar{X} - \bar{Y}) = \sqrt{\sigma_y^2/12 + \sigma_x^2/12}$ and that we estimate the population standard deviations σ_x and σ_y with the sample estimates. What is the estimate of $\mathrm{SE}(\bar{X} - \bar{Y}) = \sqrt{\sigma_y^2/12 + \sigma_x^2/12}$?
9. So now we can compute $\bar{Y} - \bar{X}$ as well as an estimate of this standard error and construct a t-statistic. What is this t-statistic?
10. If we apply the CLT, what is the distribution of this t-statistic?
    - A) Normal with mean 0 and standard deviation 1.

- B) t-distributed with 22 degrees of freedom.
- C) Normal with mean 0 and standard deviation $\sqrt{\sigma_y^2/12 + \sigma_x^2/12}$.
- D) t-distributed with 12 degrees of freedom.

11. Now we are ready to compute a p-value using the CLT. What is the probability of observing a quantity as large as what we computed in 10, when the null distribution is true?
12. CLT provides an approximation for cases in which the sample size is large. In practice, we can't check the assumption because we only get to see 1 outcome (which you computed above). As a result, if this approximation is off, so is our p-value. As described earlier, there is another approach that does not require a large sample size, but rather that the distribution of the population is approximately normal. We don't get to see this distribution so it is again an assumption, although we can look at the distribution of the sample with `qqnorm(X)` and `qqnorm(Y)`. If we are willing to assume this, then it follows that the t-statistic follows t-distribution. What is the p-value under the t-distribution approximation? Hint: use the `t.test` function.
13. With the CLT distribution, we obtained a p-value smaller than 0.05 and with the t-distribution, one that is larger. They can't both be right. What best describes the difference?
    - A) A sample size of 12 is not large enough, so we have to use the t-distribution approximation.
    - B) These are two different assumptions. The t-distribution accounts for the variability introduced by the estimation of the standard error and thus, under the null, large values are more probable under the null distribution.
    - C) The population data is probably not normally distributed so the t-distribution approximation is wrong.
    - D) Neither assumption is useful. Both are wrong.

t-tests in Practice

The R markdown document for this section is available here[35].

Introduction

We will now demonstrate how to obtain a p-value in practice. We begin by loading experimental data and walking you through the steps used to form a t-statistic and compute a p-value. We can perform this task with just a few lines of code (go to the end of section to see them). However, to understand the concepts, we will construct a t-statistic from "scratch".

Read in and prepare data

We start by reading in the data. A first important step is to identify which rows are associated with treatment and control, and to compute the difference in mean.

[35] https://github.com/genomicsclass/labs/tree/master/inference/t-tests_in_practice.Rmd

```
library(dplyr)
dat <- read.csv("femaleMiceWeights.csv") #previously downloaded

control <- filter(dat,Diet=="chow") %>% select(Bodyweight) %>% unlist
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight) %>% unlist

diff <- mean(treatment) - mean(control)
print(diff)
```

```
## [1] 3.020833
```

We are asked to report a p-value. What do we do? We learned that `diff`, referred to as the *observed effect size*, is a random variable. We also learned that under the null hypothesis, the mean of the distribution of `diff` is 0. What about the standard error? We also learned that the standard error of this random variable is the population standard deviation divided by the square root of the sample size:

$$SE(\bar{X}) = \sigma/\sqrt{N}$$

We use the sample standard deviation as an estimate of the population standard deviation. In R, we simply use the `sd` function and the SE is:

```
sd(control)/sqrt(length(control))
```

```
## [1] 0.8725323
```

This is the SE of the sample average, but we actually want the SE of `diff`. We saw how statistical theory tells us that the variance of the difference of two random variables is the sum of its variances, so we compute the variance and take the square root:

```
se <- sqrt(
  var(treatment)/length(treatment) +
  var(control)/length(control)
  )
```

Statistical theory tells us that if we divide a random variable by its SE, we get a new random variable with an SE of 1.

```
tstat <- diff/se
```

This ratio is what we call the t-statistic. It's the ratio of two random variables and thus a random variable. Once we know the distribution of this random variable, we can then easily compute a p-value.

As explained in the previous section, the CLT tells us that for large sample sizes, both sample averages `mean(treatment)` and `mean(control)` are normal. Statistical theory tells us that the difference of two normally distributed random variables is again normal, so CLT tells us that `tstat` is approximately normal with mean 0 (the null hypothesis) and SD 1 (we divided by its SE).

So now to calculate a p-value all we need to do is ask: how often does a normally distributed random variable exceed `diff`? R has a built-in function, `pnorm`, to answer this specific question. `pnorm(a)` returns the probability that a random variable following the standard normal distribution falls below `a`. To obtain the probability that it is larger than `a`, we simply use `1-pnorm(a)`. We want to know the probability of seeing something as extreme as `diff`: either smaller (more negative) than `-abs(diff)` or larger than `abs(diff)`. We call these two regions "tails" and calculate their size:

```
righttail <- 1 - pnorm(abs(tstat))
lefttail <- pnorm(-abs(tstat))
pval <- lefttail + righttail
print(pval)
```

```
## [1] 0.0398622
```

In this case, the p-value is smaller than 0.05 and using the conventional cutoff of 0.05, we would call the difference *statistically significant*.

Now there is a problem. CLT works for large samples, but is 12 large enough? A rule of thumb for CLT is that 30 is a large enough sample size (but this is just a rule of thumb). The p-value we computed is only a valid approximation if the assumptions hold, which do not seem to be the case here. However, there is another option other than using CLT.

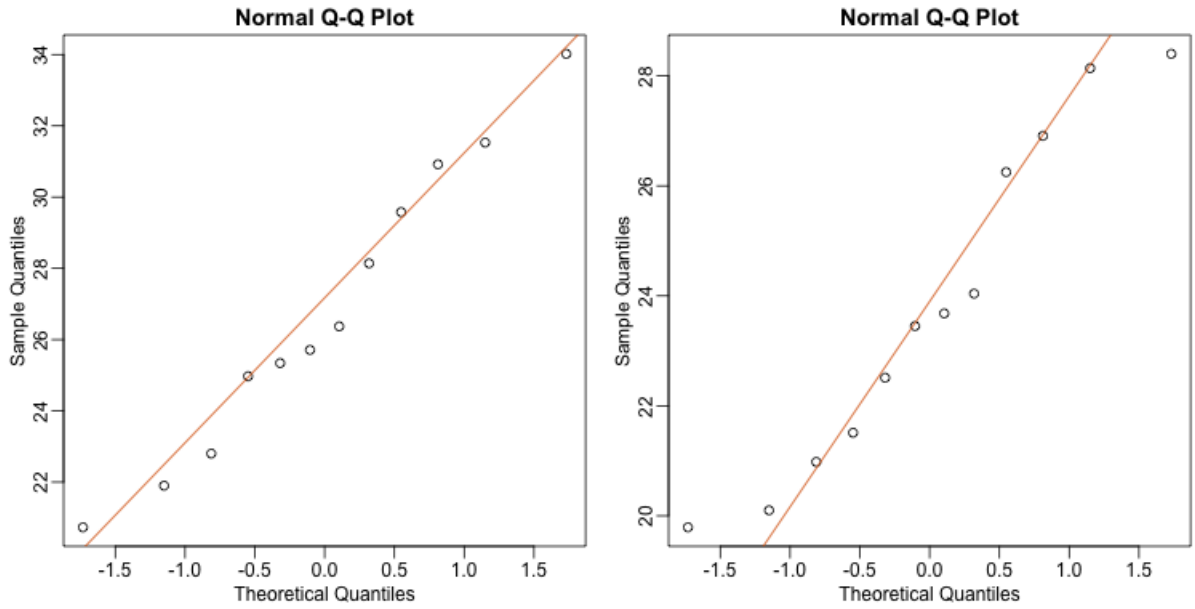
The t-distribution in Practice

The R markdown document for this section is available here[36].

As described earlier, statistical theory offers another useful result. If the distribution of the population is normal, then we can work out the exact distribution of the t-statistic without the need for the CLT. This is a big "if" given that, with small samples, it is hard to check if the population is normal. But for something like weight, we suspect that the population distribution is likely well approximated by normal and that we can use this approximation. Furthermore, we can look at a qq-plot for the samples. This shows that the approximation is at least close:

[36] https://github.com/genomicsclass/labs/tree/master/inference/t-tests_in_practice.Rmd

```
library(rafalib)
mypar(1,2)

qqnorm(treatment)
qqline(treatment,col=2)

qqnorm(control)
qqline(control,col=2)
```



Quantile-quantile plots for sample against theoretical normal distribution.

If we use this approximation, then statistical theory tells us that the distribution of the random variable `tstat` follows a t-distribution. This is a much more complicated distribution than the normal. The t-distribution has a location parameter like the normal and another parameter called *degrees of freedom*. R has a nice function that actually computes everything for us.

```
t.test(treatment, control)
```

```
##
##         Welch Two Sample t-test
##
## data:  treatment and control
## t = 2.0552, df = 20.236, p-value = 0.053
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.04296563  6.08463229
## sample estimates:
## mean of x mean of y
##  26.83417  23.81333
```

To see just the p-value, we can use the $ extractor:

```
result <- t.test(treatment,control)
result$p.value
```

```
## [1] 0.05299888
```

The p-value is slightly bigger now. This is to be expected because our CLT approximation considered the denominator of `tstat` practically fixed (with large samples it practically is), while the t-distribution approximation takes into account that the denominator (the standard error of the difference) is a random variable. The smaller the sample size, the more the denominator varies.

It may be confusing that one approximation gave us one p-value and another gave us another, because we expect there to be just one answer. However, this is not uncommon in data analysis. We used different assumptions, different approximations, and therefore we obtained different results.

Later, in the power calculation section, we will describe type I and type II errors. As a preview, we will point out that the test based on the CLT approximation is more likely to incorrectly reject the null hypothesis (a false positive), while the t-distribution is more likely to incorrectly accept the null hypothesis (false negative).

Running the t-test in practice

Now that we have gone over the concepts, we can show the relatively simple code that one would use to actually compute a t-test:

```
library(dplyr)
dat <- read.csv("mice_pheno.csv")
control <- filter(dat,Diet=="chow") %>% select(Bodyweight)
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight)
t.test(treatment,control)
```

```
##
##         Welch Two Sample t-test
##
## data:  treatment and control
## t = 7.1932, df = 735.02, p-value = 1.563e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.231533 3.906857
## sample estimates:
## mean of x mean of y
##  30.48201  27.41281
```

The arguments to `t.test` can be of type *data.frame* and thus we do not need to unlist them into numeric objects.

Confidence Intervals

The R markdown document for this section is available here[37].

We have described how to compute p-values which are ubiquitous in the life sciences. However, we do not recommend reporting p-values as the only statistical summary of your results. The reason is simple: statistical significance does not guarantee scientific significance. With large enough sample sizes, one might detect a statistically significance difference in weight of, say, 1 microgram. But is this an important finding? Would we say a diet results in higher weight if the increase is less than a fraction of a percent? The problem with reporting only p-values is that you will not provide a very important piece of information: the effect size. Recall that the effect size is the observed difference. Sometimes the effect size is divided by the mean of the control group and so expressed as a percent increase.

A much more attractive alternative is to report confidence intervals. A confidence interval includes information about your estimated effect size and the uncertainty associated with this estimate. Here we use the mice data to illustrate the concept behind confidence intervals.

[37] https://github.com/genomicsclass/labs/tree/master/inference/confidence_intervals.Rmd

Confidence Interval For Population Mean

Before we show how to construct a confidence interval for the difference between the two groups, we will show how to construct a confidence interval for the population mean of control female mice. Then we will return to the group difference after we've learned how to build confidence intervals in the simple case. We start by reading in the data and selecting the appropriate rows:

```
dat <- read.csv("mice_pheno.csv")
chowPopulation <- dat[dat$Sex=="F" & dat$Diet=="chow",3]
```

The population average μ_X is our parameter of interest here:

```
mu_chow <- mean(chowPopulation)
print(mu_chow)
```

```
## [1] 23.89338
```

We are interested in estimating this parameter. In practice, we do not get to see the entire population so, as we did for p-values, we demonstrate how we can use samples to do this. Let's start with a sample of size 30:

```
N <- 30
chow <- sample(chowPopulation,N)
print(mean(chow))
```

```
## [1] 23.351
```

We know this is a random variable, so the sample average will not be a perfect estimate. In fact, because in this illustrative example we know the value of the parameter, we can see that they are not exactly the same. A confidence interval is a statistical way of reporting our finding, the sample average, in a way that explicitly summarizes the variability of our random variable.

With a sample size of 30, we will use the CLT. The CLT tells us that $\bar{X}$ or `mean(chow)` follows a normal distribution with mean μ_X or `mean(chowPopulation)` and standard error approximately $s_X/\sqrt{N}$ or:

```
se <- sd(chow)/sqrt(N)
print(se)
```

```
## [1] 0.4781652
```

Defining The Interval

A 95% confidence interval (we can use percentages other than 95%) is a random interval with a 95% probability of falling on the parameter we are estimating. Keep in mind that saying 95% of random intervals will fall on the true value (our definition above) is *not the same* as saying there is a 95% chance that the true value falls in our interval. To construct it, we note that the CLT tells us that $\sqrt{N}(\bar{X} - \mu_X)/s_X$ follows a normal distribution with mean 0 and SD 1. This implies that the probability of this event:

$$-2 \leq \sqrt{N}(\bar{X} - \mu_X)/s_X \leq 2$$

which written in R code is:

```
pnorm(2) - pnorm(-2)
```

```
## [1] 0.9544997
```

...is about 95% (to get closer use `qnorm(1-0.05/2)` instead of 2). Now do some basic algebra to clear out everything and leave μ_X alone in the middle and you get that the following event:

$$\bar{X} - 2s_X/\sqrt{N} \leq \mu_X \leq \bar{X} + 2s_X/\sqrt{N}$$

has a probability of 95%.

Be aware that it is the edges of the interval $\bar{X} \pm 2s_X/\sqrt{N}$, not μ_X , that are random. Again, the definition of the confidence interval is that 95% of *random intervals* will contain the true, fixed value μ_X. For a specific interval that has been calculated, the probability is either 0 or 1 that it contains the fixed population mean μ_X.

Let's demonstrate this logic through simulation. We can construct this interval with R relatively easily:

```
Q <- qnorm(1- 0.05/2)
interval <- c(mean(chow)-Q*se, mean(chow)+Q*se )
interval
```

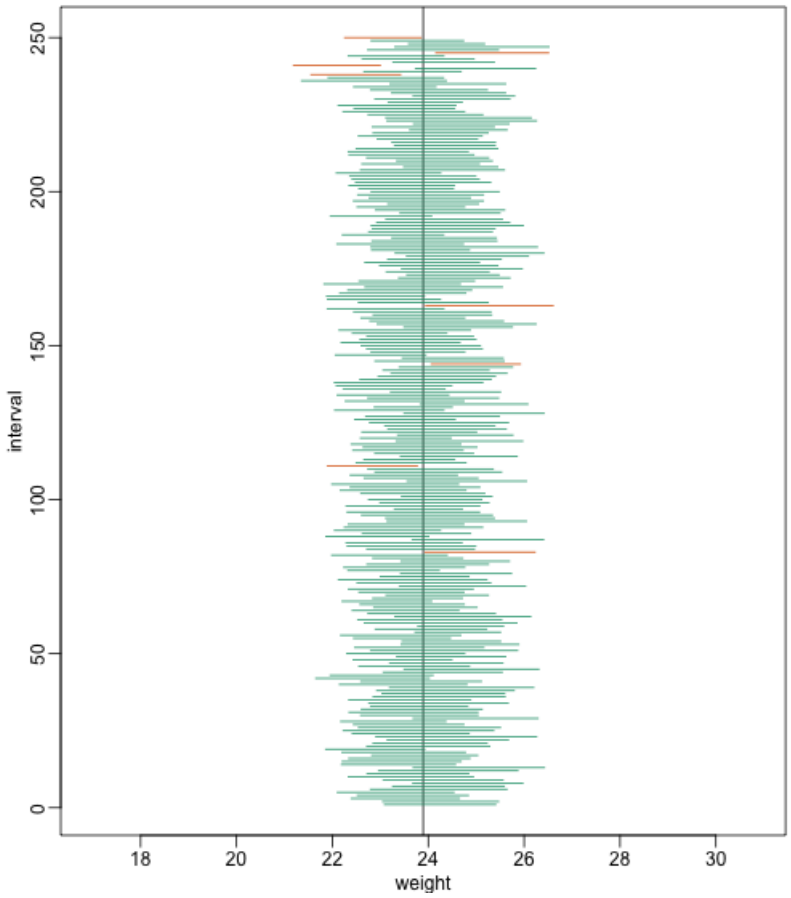
```
## [1] 22.41381 24.28819
```

```
interval[1] < mu_chow & interval[2] > mu_chow
```

```
## [1] TRUE
```

which happens to cover μ_X or `mean(chowPopulation)`. However, we can take another sample and we might not be as lucky. In fact, the theory tells us that we will cover μ_X 95% of the time. Because we have access to the population data, we can confirm this by taking several new samples:

```
library(rafalib)
B <- 250
mypar()
plot(mean(chowPopulation)+c(-7,7),c(1,1),type="n",
     xlab="weight",ylab="interval",ylim=c(1,B))
abline(v=mean(chowPopulation))
for (i in 1:B) {
  chow <- sample(chowPopulation,N)
  se <- sd(chow)/sqrt(N)
  interval <- c(mean(chow)-Q*se, mean(chow)+Q*se)
  covered <-
    mean(chowPopulation) <= interval[2] & mean(chowPopulation) >= interval[1]
  color <- ifelse(covered,1,2)
  lines(interval, c(i,i),col=color)
}
```



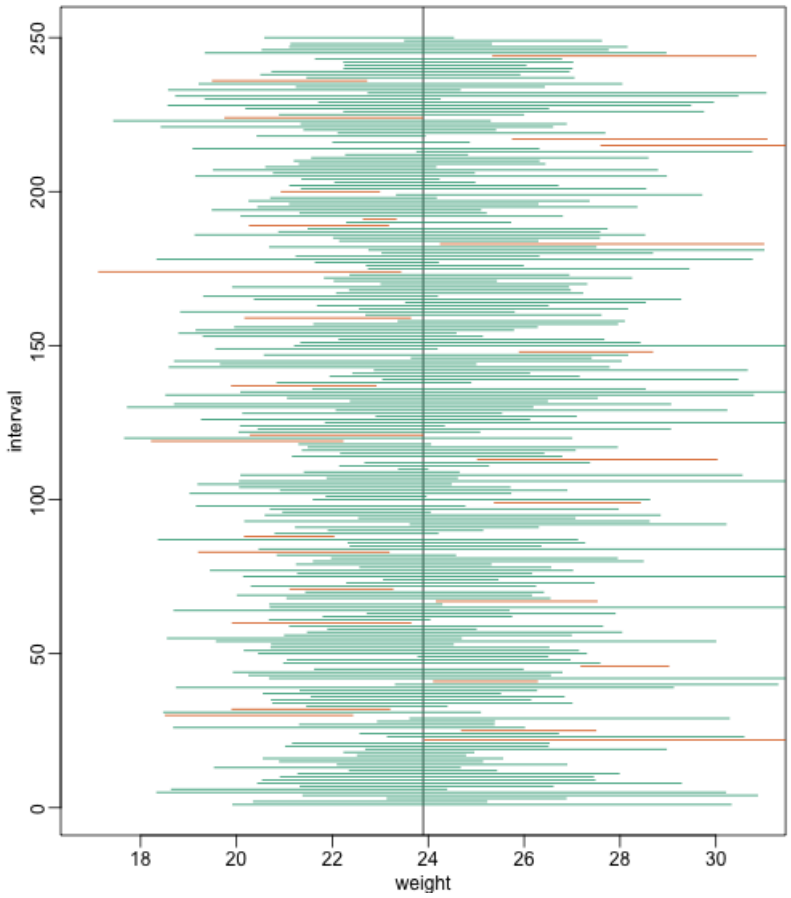
We show 250 random realizations of 95% confidence intervals. The color denotes if the interval fell on the parameter or not.

You can run this repeatedly to see what happens. You will see that in about 5% of the cases, we fail to cover μ_X.

Small Sample Size And The CLT

For $N = 30$, the CLT works very well. However, if $N = 5$, do these confidence intervals work as well? We used the CLT to create our intervals, and with $N = 5$ it may not be as useful an approximation. We can confirm this with a simulation:

```
mypar()
plot(mean(chowPopulation)+c(-7,7),c(1,1),type="n",
     xlab="weight",ylab="interval",ylim=c(1,B))
abline(v=mean(chowPopulation))
Q <- qnorm(1- 0.05/2)
N <- 5
for (i in 1:B) {
  chow <- sample(chowPopulation,N)
  se <- sd(chow)/sqrt(N)
  interval <- c(mean(chow)-Q*se, mean(chow)+Q*se)
  covered <- mean(chowPopulation) <= interval[2] & mean(chowPopulation) >= inter\
val[1]
  color <- ifelse(covered,1,2)
  lines(interval, c(i,i),col=color)
}
```

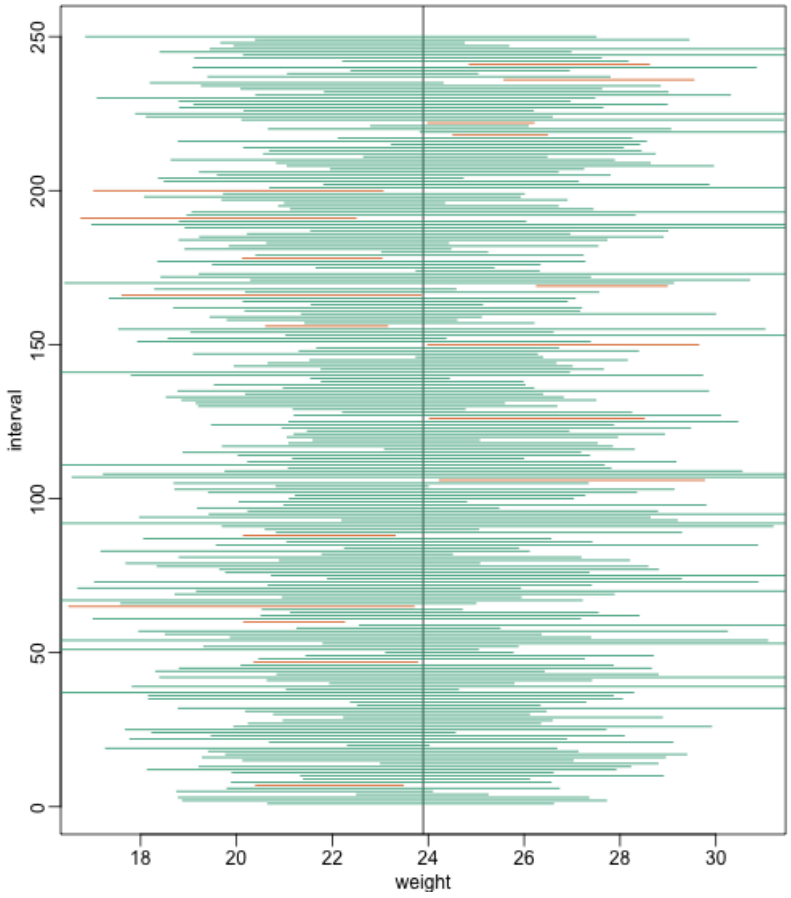


We show 250 random realizations of 95% confidence intervals, but now for a smaller sample size. The confidence interval is based on the CLT approximation. The color denotes if the interval fell on the parameter or not.

Despite the intervals being larger (we are dividing by $\sqrt{5}$ instead of $\sqrt{30}$), we see many more intervals not covering μ_X. This is because the CLT is incorrectly telling us that the distribution of the `mean(chow)` is approximately normal when, in fact, it has a fatter tail (the parts of the distribution going to $\pm\infty$). This mistake affects us in the calculation of `Q`, which assumes a normal distribution and uses `qnorm`. The t-distribution might be more appropriate. All we have to do is re-run the above, but change how we calculate `Q` to use `qt` instead of `qnorm`.

```
mypar()
plot(mean(chowPopulation) + c(-7,7), c(1,1), type="n",
     xlab="weight", ylab="interval", ylim=c(1,B))
abline(v=mean(chowPopulation))
##Q <- qnorm(1- 0.05/2) ##no longer normal so use:
Q <- qt(1- 0.05/2, df=4)
N <- 5
for (i in 1:B) {
  chow <- sample(chowPopulation, N)
```

```
  se <- sd(chow)/sqrt(N)
  interval <- c(mean(chow)-Q*se, mean(chow)+Q*se )
  covered <- mean(chowPopulation) <= interval[2] & mean(chowPopulation) >= inter\
val[1]
  color <- ifelse(covered,1,2)
  lines(interval, c(i,i),col=color)
}
```



We show 250 random realizations of 95% confidence intervals, but now for a smaller sample size. The confidence is now based on the t-distribution approximation. The color denotes if the interval fell on the parameter or not.

Now the intervals are made bigger. This is because the t-distribution has fatter tails and therefore:

```
qt(1- 0.05/2, df=4)
```

```
## [1] 2.776445
```

is bigger than...

```
qnorm(1- 0.05/2)
```

```
## [1] 1.959964
```

...which makes the intervals larger and hence cover μ_X more frequently; in fact, about 95% of the time.

Connection Between Confidence Intervals and p-values

We recommend that in practice confidence intervals be reported instead of p-values. If for some reason you are required to provide p-values, or required that your results are significant at the 0.05 of 0.01 levels, confidence intervals do provide this information.

If we are talking about a t-test p-value, we are asking if differences as extreme as the one we observe, $\bar{Y} - \bar{X}$, are likely when the difference between the population averages is actually equal to zero. So we can form a confidence interval with the observed difference. Instead of writing $\bar{Y} - \bar{X}$ repeatedly, let's define this difference as a new variable $d \equiv \bar{Y} - \bar{X}$.

Suppose you use CLT and report $d \pm 2s_d/\sqrt{N}$ as a 95% confidence interval for the difference and this interval does not include 0 (a false positive). Because the interval does not include 0, this implies that either $D - 2s_d/\sqrt{N} > 0$ or $d + 2s_d/\sqrt{N} < 0$. This suggests that either $\sqrt{N}d/s_d > 2$ or $\sqrt{N}d/s_d < 2$. This then implies that the t-statistic is more extreme than 2, which in turn suggests that the p-value must be smaller than 0.05 (approximately, for a more exact calculation use `qnorm(.05/2)` instead of 2). The same calculation can be made if we use the t-distribution instead of CLT (with `qt(.05/2, df=N-2)`). In summary, if a 95% or 99% confidence interval does not include 0, then the p-value must be smaller than 0.05 or 0.01 respectively.

Note that the confidence interval for the difference d is provided by the `t.test` function:

```
## [1] -0.04296563  6.08463229
## attr(,"conf.level")
## [1] 0.95
```

In this case, the 95% confidence interval does include 0 and we observe that the p-value is larger than 0.05 as predicted. If we change this to a 90% confidence interval, then:

```
t.test(treatment,control,conf.level=0.9)$conf.int
```

```
## [1] 0.4871597 5.5545070
## attr(,"conf.level")
## [1] 0.9
```

0 is no longer in the confidence interval (which is expected because the p-value is smaller than 0.10).

Power Calculations

The R markdown document for this section is available here[38].

Introduction

We have used the example of the effects of two different diets on the weight of mice. Since in this illustrative example we have access to the population, we know that in fact there is a substantial (about 10%) difference between the average weights of the two populations:

```
library(dplyr)
dat <- read.csv("mice_pheno.csv") #Previously downloaded

controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
  select(Bodyweight) %>% unlist

hfPopulation <- filter(dat,Sex == "F" & Diet == "hf") %>%
  select(Bodyweight) %>% unlist

mu_hf <- mean(hfPopulation)
mu_control <- mean(controlPopulation)
print(mu_hf - mu_control)
```

```
## [1] 2.375517
```

```
print((mu_hf - mu_control)/mu_control * 100) # percent increase
```

```
## [1] 9.942157
```

We have also seen that, in some cases, when we take a sample and perform a t-test, we don't always get a p-value smaller than 0.05. For example, here is a case were we take sample of 5 mice and don't achieve statistical significance at the 0.05 level:

[38] https://github.com/genomicsclass/labs/tree/master/inference/power_calculations.Rmd

```
set.seed(1)
N <- 5
hf <- sample(hfPopulation,N)
control <- sample(controlPopulation,N)
t.test(hf,control)$p.value
```

```
## [1] 0.1410204
```

Did we make a mistake? By not rejecting the null hypothesis, are we saying the diet has no effect? The answer to this question is no. All we can say is that we did not reject the null hypothesis. But this does not necessarily imply that the null is true. The problem is that, in this particular instance, we don't have enough *power*, a term we are now going to define. If you are doing scientific research, it is very likely that you will have to do a power calculation at some point. In many cases, it is an ethical obligation as it can help you avoid sacrificing mice unnecessarily or limiting the number of human subjects exposed to potential risk in a study. Here we explain what statistical power means.

Types Of Error

Whenever we perform a statistical test, we are aware that we may make a mistake. This is why our p-values are not 0. Under the null, there is always a positive, perhaps very small, but still positive chance that we will reject the null when it is true. If the p-value is 0.05, it will happen 1 out of 20 times. This *error* is called *type I error* by statisticians.

A type I error is defined as rejecting the null when we should not. This is also referred to as a false positive. So why do we then use 0.05? Shouldn't we use 0.000001 to be really sure? The reason we don't use infinitesimal cut-offs to avoid type I errors at all cost is that there is another error we can commit: to not reject the null when we should. This is called a *type II error* or a false negative. The R code analysis above shows an example of a false negative: we did not reject the null hypothesis (at the 0.05 level) and, because we happen to know and peeked at the true population means, we know there is in fact a difference. Had we used a p-value cutoff of 0.25, we would not have made this mistake. However, in general, are we comfortable with a type I error rate of 1 in 4? Usually we are not.

The 0.05 and 0.01 Cut-offs Are Arbitrary

Most journals and regulatory agencies frequently insist that results be significant at the 0.01 or 0.05 levels. Of course there is nothing special about these numbers other than the fact that some of the first papers on p-values used these values as examples. Part of the goal of this book is to give readers a good understanding of what p-values and confidence intervals are so that these choices can be judged in an informed way. Unfortunately, in science, these cut-offs are applied somewhat mindlessly, but that topic is part of a complicated debate.

Power Calculation

Power is the probability of rejecting the null when the null is false. Of course "when the null is false" is a complicated statement because it can be false in many ways. $\Delta \equiv \mu_Y - \mu_X$ could be anything and the power actually depends on this parameter. It also depends on the standard error of your estimates which in turn depends on the sample size and the population standard deviations. In practice, we don't know these so we usually report power for several plausible values of Δ, σ_X, σ_Y and various sample sizes. Statistical theory gives us formulas to calculate power. The `pwr` package performs these calculations for you. Here we will illustrate the concepts behind power by coding up simulations in R.

Suppose our sample size is:

```
N <- 12
```

and we will reject the null hypothesis at:

```
alpha <- 0.05
```

What is our power with this particular data? We will compute this probability by re-running the exercise many times and calculating the proportion of times the null hypothesis is rejected. Specifically, we will run:

```
B <- 2000
```

simulations. The simulation is as follows: we take a sample of size N from both control and treatment groups, we perform a t-test comparing these two, and report if the p-value is less than `alpha` or not. We write a function that does this:

```
reject <- function(N, alpha=0.05){
   hf <- sample(hfPopulation,N)
   control <- sample(controlPopulation,N)
   pval <- t.test(hf,control)$p.value
   pval < alpha
}
```

Here is an example of one simulation for a sample size of 12. The call to `reject` answers the question "Did we reject?"

```
reject(12)
```

```
## [1] FALSE
```

Now we can use the `replicate` function to do this `B` times.

```
rejections <- replicate(B,reject(N))
```

Our power is just the proportion of times we correctly reject. So with $N = 12$ our power is only:

```
mean(rejections)
```

```
## [1] 0.2145
```

This explains why the t-test was not rejecting when we knew the null was false. With a sample size of just 12, our power is about 23%. To guard against false positives at the 0.05 level, we had set the threshold at a high enough level that resulted in many type II errors.

Let's see how power improves with N. We will use the function `sapply`, which applies a function to each of the elements of a vector. We want to repeat the above for the following sample size:

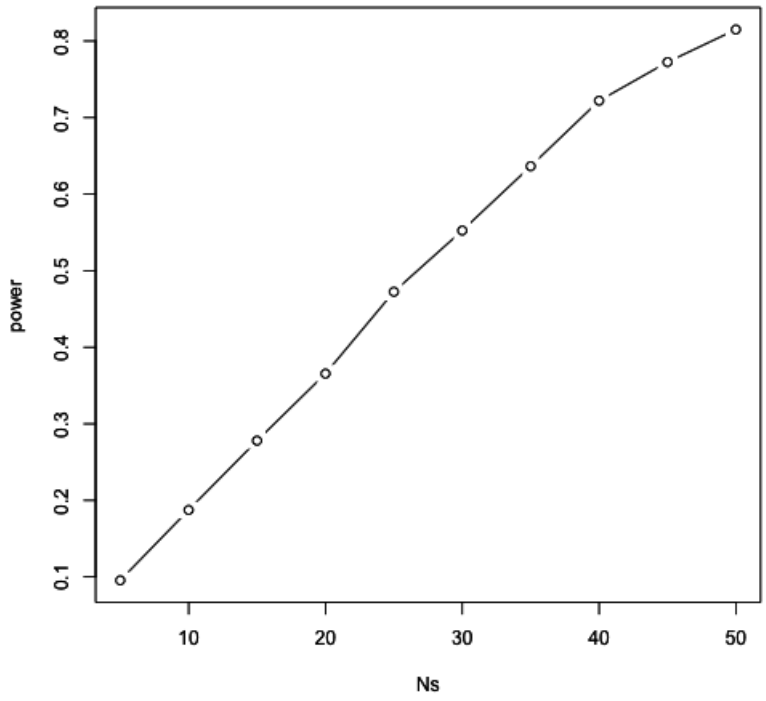
```
Ns <- seq(5, 50, 5)
```

So we use `apply` like this:

```
power <- sapply(Ns,function(N){
  rejections <- replicate(B, reject(N))
  mean(rejections)
  })
```

For each of the three simulations, the above code returns the proportion of times we reject. Not surprisingly power increases with N:

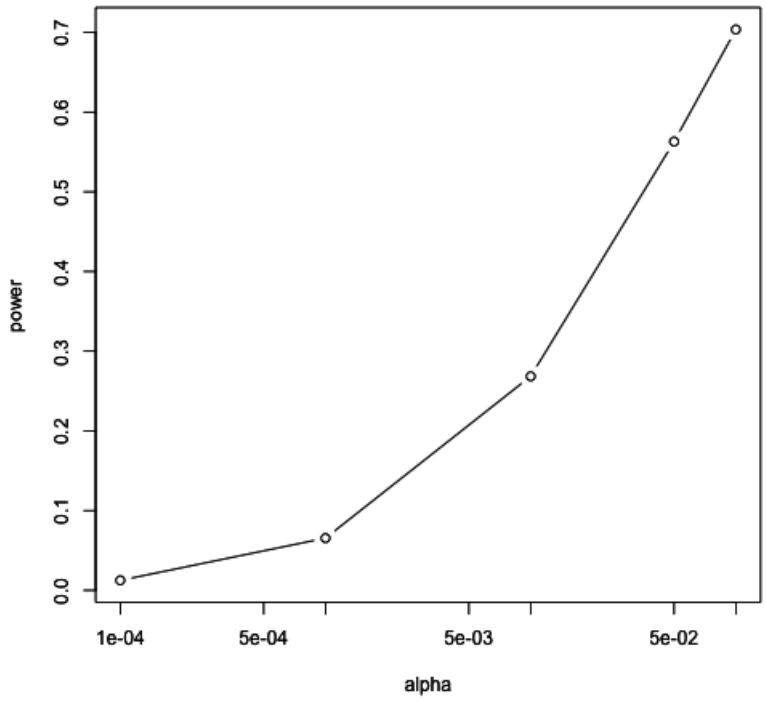
```
plot(Ns, power, type="b")
```



Power plotted against sample size.

Similarly, if we change the level `alpha` at which we reject, power changes. The smaller I want the chance of type I error to be, the less power I will have. Another way of saying this is that we trade off between the two types of error. We can see this by writing similar code, but keeping N fixed and considering several values of `alpha`:

```
N <- 30
alphas <- c(0.1,0.05,0.01,0.001,0.0001)
power <- sapply(alphas,function(alpha){
  rejections <- replicate(B,reject(N,alpha=alpha))
  mean(rejections)
})
plot(alphas, power, xlab="alpha", type="b", log="x")
```



Power plotted against cut-off.

Note that the x-axis in this last plot is in the log scale.

There is no "right" power or "right" alpha level, but it is important that you understand what each means.

To see this clearly, you could create a plot with curves of power versus N. Show several curves in the same plot with color representing alpha level.

p-values are Arbitrary under the Alternative Hypothesis

Another consequence of what we have learned about power is that p-values are somewhat arbitrary when the null hypothesis is not true and therefore the *alternative* hypothesis is true (the difference between the population means is not zero). When the alternative hypothesis is true, we can make a p-value as small as we want simply by increasing the sample size (supposing that we have an infinite population to sample from). We can show this property of p-values by drawing larger and larger samples from our population and calculating p-values. This works because, in our case, we know that the alternative hypothesis is true, since we have access to the populations and can calculate the difference in their means.

First write a function that returns a p-value for a given sample size N:

```
calculatePvalue <- function(N) {
   hf <- sample(hfPopulation,N)
   control <- sample(controlPopulation,N)
   t.test(hf,control)$p.value
}
```

We have a limit here of 200 for the high-fat diet population, but we can see the effect well before we get to 200. For each sample size, we will calculate a few p-values. We can do this by repeating each value of N a few times.

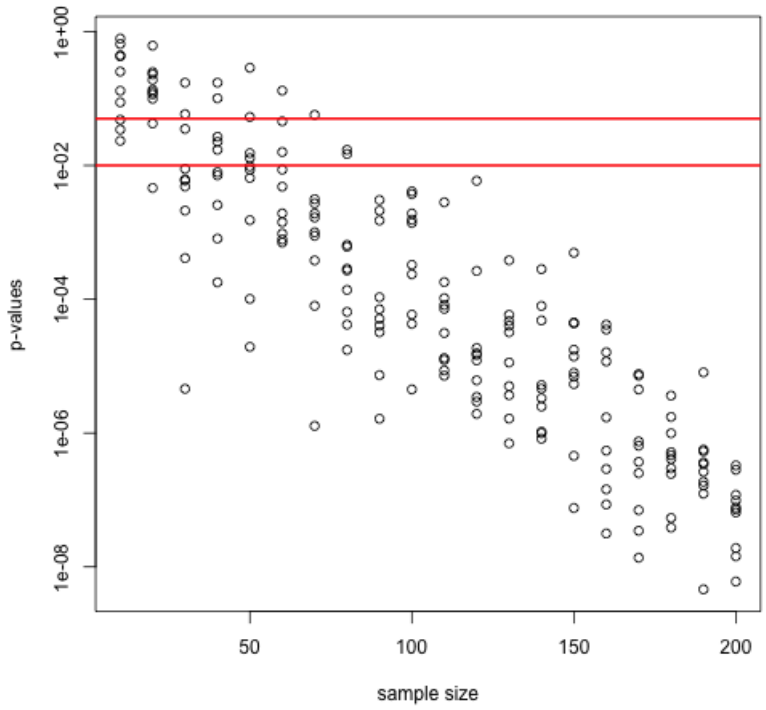
```
Ns <- seq(10,200,by=10)
Ns_rep <- rep(Ns, each=10)
```

Again we use `sapply` to run our simulations:

```
pvalues <- sapply(Ns_rep, calculatePvalue)
```

Now we can plot the 10 p-values we generated for each sample size:

```
plot(Ns_rep, pvalues, log="y", xlab="sample size",
     ylab="p-values")
abline(h=c(.01, .05), col="red", lwd=2)
```



p-values from random samples at varying sample size. The actual value of the p-values decreases as we increase sample size whenever the alternative hypothesis is true.

Note that the y-axis is log scale and that the p-values show a decreasing trend all the way to 10^{-8} as the sample size gets larger. The standard cutoffs of 0.01 and 0.05 are indicated with horizontal red lines.

It is important to remember that p-values are not more interesting as they become very very small. Once we have convinced ourselves to reject the null hypothesis at a threshold we find reasonable, having an even smaller p-value just means that we sampled more mice than was necessary. Having a larger sample size does help to increase the precision of our estimate of the difference Δ, but the fact that the p-value becomes very very small is just a natural consequence of the mathematics of the test. The p-values get smaller and smaller with increasing sample size because the numerator of the t-statistic has $\sqrt{N}$ (for equal sized groups, and a similar effect occurs when $M \neq N$). Therefore, if Δ is non-zero, the t-statistic will increase with N.

Therefore, a better statistic to report is the effect size with a confidence interval or some statistic which gives the reader a sense of the change in a meaningful scale. We can report the effect size as a percent by dividing the difference and the confidence interval by the control population mean:

```
N <- 12
hf <- sample(hfPopulation, N)
control <- sample(controlPopulation, N)
diff <- mean(hf) - mean(control)
diff / mean(control) * 100
```

```
## [1] 1.868663
```

```
t.test(hf, control)$conf.int / mean(control) * 100
```

```
## [1] -20.94576  24.68309
## attr(,"conf.level")
## [1] 0.95
```

In addition, we can report a statistic called Cohen's d[39], which is the difference between the groups divided by the pooled standard deviation of the two groups.

```
sd_pool <- sqrt(((N-1)*var(hf) + (N-1)*var(control))/(2*N - 2))
diff / sd_pool
```

```
## [1] 0.07140083
```

This tells us how many standard deviations of the data the mean of the high-fat diet group is from the control group. Under the alternative hypothesis, unlike the t-statistic which is guaranteed to increase, the effect size and Cohen's d will become more precise.

Exercises

For these exercises we will load the babies dataset from `babies.txt`. We will use this data to review the concepts behind the p-values and then test confidence interval concepts.

```
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/babies.txt"
filename <- basename(url)
download(url, destfile=filename)
babies <- read.table("babies.txt", header=TRUE)
```

This is a large dataset (1,236 cases), and we will pretend that it contains the entire population in which we are interested. We will study the differences in birth weight between babies born to

[39] https://en.wikipedia.org/wiki/Effect_size#Cohen.27s_d

smoking and non-smoking mothers.

First, let's split this into two birth weight datasets: one of birth weights to non-smoking mothers and the other of birth weights to smoking mothers.

```
bwt.nonsmoke <- filter(babies, smoke==0) %>% select(bwt) %>% unlist
bwt.smoke <- filter(babies, smoke==1) %>% select(bwt) %>% unlist
```

Now, we can look for the true population difference in means between smoking and non-smoking birth weights.

```
library(rafalib)
mean(bwt.nonsmoke)-mean(bwt.smoke)
popsd(bwt.nonsmoke)
popsd(bwt.smoke)
```

The population difference of mean birth weights is about 8.9 ounces. The standard deviations of the nonsmoking and smoking groups are about 17.4 and 18.1 ounces, respectively.

As we did with the mouse weight data, this assessment interactively reviews inference concepts using simulations in R. We will treat the babies dataset as the full population and draw samples from it to simulate individual experiments. We will then ask whether somebody who only received the random samples would be able to draw correct conclusions about the population.

We are interested in testing whether the birth weights of babies born to non-smoking mothers are significantly different from the birth weights of babies born to smoking mothers.

1. Set the seed at 1 and obtain two samples, each of size $N = 25$, from non-smoking mothers (`dat.ns`) and smoking mothers (`dat.s`). Compute the t-statistic (call it `tval`).
2. Recall that we summarize our data using a t-statistics because we know that in situations where the null hypothesis is true (what we mean when we say "under the null") and the sample size is relatively large, this t-value will have an approximate standard normal distribution. Because we know the distribution of the t-value under the null, we can quantitatively determine how unusual the observed t-value would be if the null hypothesis were true.

   The standard procedure is to examine the probability a t-statistic that actually does follow the null hypothesis would have larger absolute value than the absolute value of the t-value we just observed – this is called a two-sided test.

   We have computed these by taking one minus the area under the standard normal curve between `-abs(tval)` and `abs(tval)`. In R, we can do this by using the `pnorm` function, which computes the area under a normal curve from negative infinity up to the value given as its first argument:

3. Because of the symmetry of the standard normal distribution, there is a simpler way to calculate the probability that a t-value under the null could have a larger absolute value than `tval`. Choose the simplified calculation from the following:
    - A) `1-2*pnorm(abs(tval))`
    - B) `1-2*pnorm(-abs(tval))`
    - C) `1-pnorm(-abs(tval))`
    - D) `2*pnorm(-abs(tval))`
4. By reporting only p-values, many scientific publications provide an incomplete story of their findings. As we have mentioned, with very large sample sizes, scientifically insignificant differences between two groups can lead to small p-values. Confidence intervals are more informative as they include the estimate itself. Our estimate of the difference between babies of smoker and non-smokers: `mean(dat.s) - mean( dat.ns)`. If we use the CLT, what quantity would we add and subtract to this estimate to obtain a 99% confidence interval?
5. If instead of CLT, we use the t-distribution approximation, what do we add and subtract (use `2*N-2` degrees of freedom)?
6. Why are the values from 4 and 5 so similar?
    - A) Coincidence.
    - B) They are both related to 99% confidence intervals.
    - C) `N` and thus the degrees of freedom is large enough to make the normal and t-distributions very similar.
    - D) They are actually quite different, differing by more than 1 ounce.
7. No matter which way you compute it, the p-value `pval` is the probability that the null hypothesis could have generated a t-statistic more extreme than than what we observed: `tval`. If the p-value is very small, this means that observing a value more extreme than `tval` would be very rare if the null hypothesis were true, and would give strong evidence that we should **reject** the null hypothesis. We determine how small the p-value needs to be to reject the null by deciding how often we would be willing to mistakenly reject the null hypothesis.

    The standard decision rule is the following: choose some small value α (in most disciplines the conventional choice is $\alpha = 0.05$) and reject the null hypothesis if the p-value is less than α. We call α the *significance level* of the test.

    It turns out that if we follow this decision rule, the probability that we will reject the null hypothesis by mistake is equal to α. (This fact is not immediately obvious and requires some probability theory to show.) We call the *event* of rejecting the null hypothesis, when it is in fact true, a *Type I error*, we call the *probability* of making a Type I error, the *Type I error rate*, and we say that rejecting the null hypothesis when the p-value is less than α, *controls* the Type I error rate so that it is equal to α. We will see a number of decision rules that we use in order to control the probabilities of other types of errors. Often, we will guarantee that the probability of an error is less than some level, but, in this case, we can guarantee that the probability of a Type I error is *exactly equal* to α.

    Which of the following sentences about a Type I error is **not** true?
    - A) The following is another way to describe a Type I error: you decided to reject the null hypothesis on the basis of data that was actually generated by the null hypothesis.

- B) The following is the another way to describe a Type I error: due to random fluctuations, even though the data you observed were actually generated by the null hypothesis, the p-value calculated from the observed data was small, so you rejected it.
- C) From the original data alone, you can tell whether you have made a Type I error.
- D) In scientific practice, a Type I error constitutes reporting a "significant" result when there is actually no result.

8. In the simulation we have set up here, we know the null hypothesis is false – the true value of difference in means is actually around 8.9. Thus, we are concerned with how often the decision rule outlined in the last section allows us to conclude that the null hypothesis is actually false. In other words, we would like to quantify the *Type II error rate* of the test, or the probability that we fail to reject the null hypothesis when the alternative hypothesis is true.

   Unlike the Type I error rate, which we can characterize by assuming that the null hypothesis of "no difference" is true, the Type II error rate cannot be computed by assuming the alternative hypothesis alone because the alternative hypothesis alone does not specify a particular value for the difference. It thus does not nail down a specific distribution for the t-value under the alternative.

   For this reason, when we study the Type II error rate of a hypothesis testing procedure, we need to assume a particular *effect size*, or hypothetical size of the difference between population means, that we wish to target. We ask questions such as "what is the smallest difference I could reliably distinguish from 0 given my sample size N?" or, more commonly, "How big does N have to be in order to detect that the absolute value of the difference is greater than zero?" Type II error control plays a major role in designing data collection procedures **before** you actually see the data, so that you know the test you will run has enough sensitivity or *power*. Power is one minus the Type II error rate, or the probability that you will reject the null hypothesis when the alternative hypothesis is true.

   There are several aspects of a hypothesis test that affect its power for a particular effect size. Intuitively, setting a lower α decreases the power of the test for a given effect size because the null hypothesis will be more difficult to reject. This means that for an experiment with fixed parameters (i.e., with a predetermined sample size, recording mechanism, etc), the power of the hypothesis test trades off with its Type I error rate, no matter what effect size you target.

   We can explore the trade off of power and Type I error concretely using the babies data. Since we have the full population, we know what the true effect size is (about 8.93) and we can compute the power of the test for true difference between populations.

   Set the seed at 1 and take a random sample of $N = 5$ measurements from each of the smoking and nonsmoking datasets. What is the p-value (use the `t-test` function)?

9. The p-value is larger than 0.05 so using the typical cut-off, we would not reject. This is a type II error. Which of the following is *not* a way to decrease this type of error?
   - A) Increase our chance of a type I error.
   - B) Take a larger sample size.

- C) Find a population for which the null is not true.
- D) Use a higher α level.

10. Set the seed at 1, then use the `replicate` function to repeat the code used in exercise 9 10,000 times. What proportion of the time do we reject at the 0.05 level?
11. Note that, not surprisingly, the power is lower than 10%. Repeat the exercise above for samples sizes of 30, 60, 90 and 120. Which of those four gives you power of about 80%?
12. Repeat problem 11, but now require an α level of 0.01. Which of those four gives you power of about 80%?

Monte Carlo Simulation

The R markdown document for this section is available here[40].

Computers can be used to generate pseudo-random numbers. For practical purposes these pseudo-random numbers can be used to imitate random variables from the real world. This permits us to examine properties of random variables using a computer instead of theoretical or analytical derivations. One very useful aspect of this concept is that we can create *simulated* data to test out ideas or competing methods, without actually having to perform laboratory experiments.

Simulations can also be used to check theoretical or analytical results. Also, many of the theoretical results we use in statistics are based on asymptotics: they hold when the sample size goes to infinity. In practice, we never have an infinite number of samples so we may want to know how well the theory works with our actual sample size. Sometimes we can answer this question analytically, but not always. Simulations are extremely useful in these cases.

As an example, let's use a Monte Carlo simulation to compare the CLT to the t-distribution approximation for different sample sizes.

```
library(dplyr)
dat <- read.csv("mice_pheno.csv")
controlPopulation <- filter(dat,Sex == "F" & Diet == "chow") %>%
  select(Bodyweight) %>% unlist
```

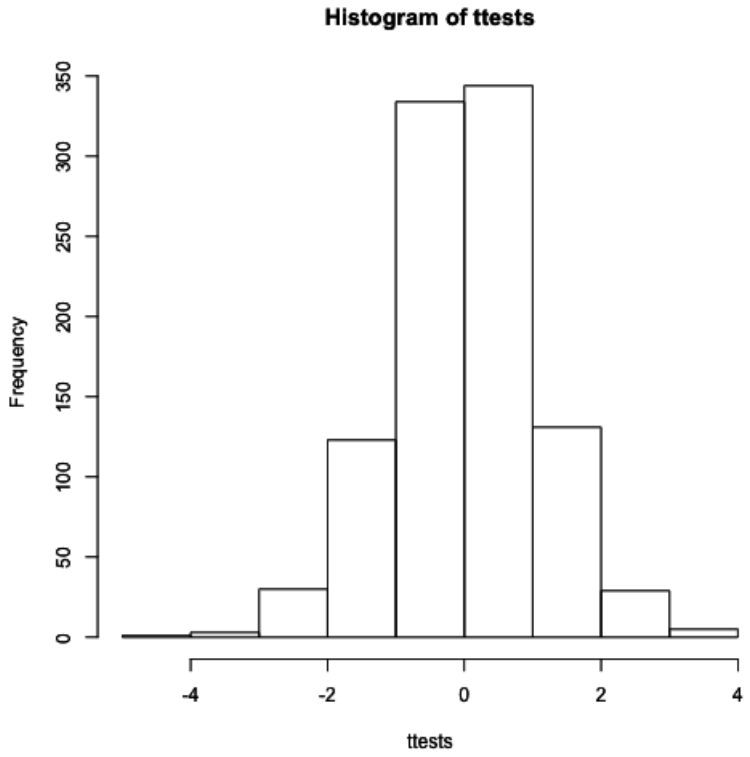
We will build a function that automatically generates a t-statistic under the null hypothesis for a any sample size of `n`.

[40] https://github.com/genomicsclass/labs/tree/master/inference/monte_carlo.Rmd

```
ttestgenerator <- function(n) {
  #note that here we have a false "high fat" group where we actually
  #sample from the nonsmokers. this is because we are modeling the *null*
  cases <- sample(controlPopulation,n)
  controls <- sample(controlPopulation,n)
  tstat <- (mean(cases)-mean(controls)) /
      sqrt( var(cases)/n + var(controls)/n )
  return(tstat)
  }
ttests <- replicate(1000, ttestgenerator(10))
```

With 1,000 Monte Carlo simulated occurrences of this random variable, we can now get a glimpse of its distribution:

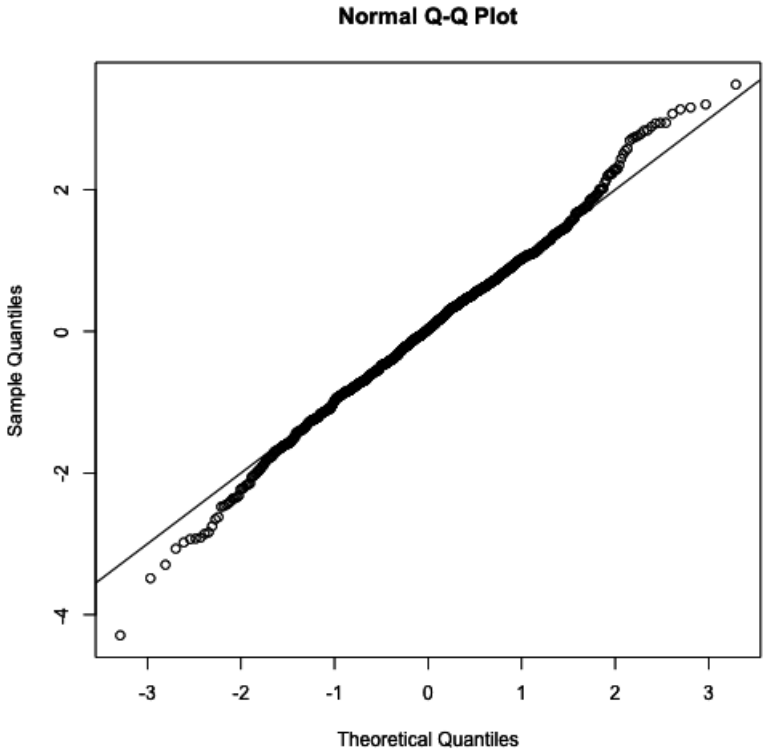
```
hist(ttests)
```



Histogram of 1000 Monte Carlo simulated t-statistics.

So is the distribution of this t-statistic well approximated by the normal distribution? In the next chapter, we will formally introduce quantile-quantile plots, which provide a useful visual inspection of how well one distribution approximates another. As we will explain later, if points fall on the identity line, it means the approximation is a good one.

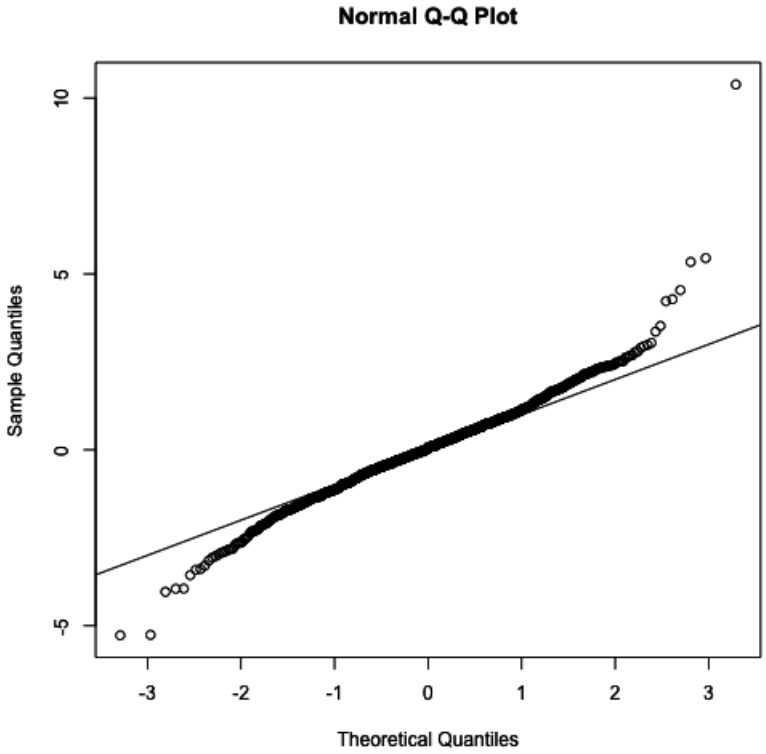
```
qqnorm(ttests)
abline(0,1)
```



Quantile-quantile plot comparing 1000 Monte Carlo simulated t-statistics to theoretical normal distribution.

This looks like a very good approximation. For this particular population, a sample size of 10 was large enough to use the CLT approximation. How about 3?

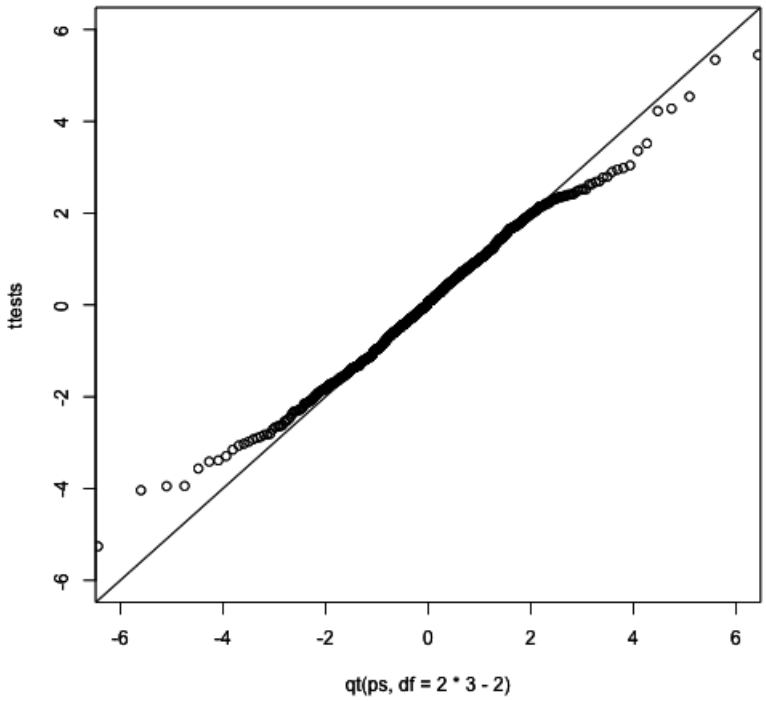
```
ttests <- replicate(1000, ttestgenerator(3))
qqnorm(ttests)
abline(0,1)
```



Quantile-quantile plot comparing 1000 Monte Carlo simulated t-statistics with three degrees of freedom to theoretical normal distribution.

Now we see that the large quantiles, referred to by statisticians as the *tails*, are larger than expected (below the line on the left side of the plot and above the line on the right side of the plot). In the previous module, we explained that when the sample size is not large enough and the *population values* follow a normal distribution, then the t-distribution is a better approximation. Our simulation results seem to confirm this:

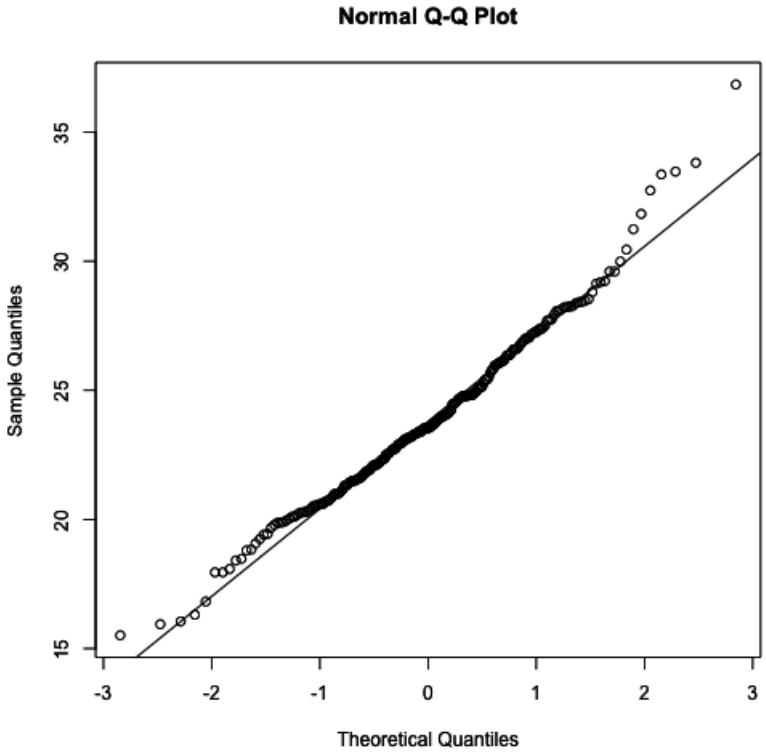
```
ps <- (seq(0,999)+0.5)/1000
qqplot(qt(ps,df=2*3-2),ttests,xlim=c(-6,6),ylim=c(-6,6))
abline(0,1)
```



Quantile-quantile plot comparing 1000 Monte Carlo simulated t-statistics with three degrees of freedom to theoretical t-distribution.

The t-distribution is a much better approximation in this case, but it is still not perfect. This is due to the fact that the original data is not that well approximated by the normal distribution.

```
qqnorm(controlPopulation)
qqline(controlPopulation)
```



Quantile-quantile of original data compared to theoretical quantile distribution.

Parametric Simulations for the Observations

The R markdown document for this section is available here[41].

The technique we used to motivate random variables and the null distribution was a type of Monte Carlo simulation. We had access to population data and generated samples at random. In practice, we do not have access to the entire population. The reason for using the approach here was for educational purposes. However, when we want to use Monte Carlo simulations in practice, it is much more typical to assume a parametric distribution and generate a population from this, which is called a *parametric simulation*. This means that we take parameters estimated from the real data (here the mean and the standard deviation), and plug these into a model (here the normal distribution). This is actually the most common form of Monte Carlo simulation.

For the case of weights, we could use our knowledge that mice typically weigh 24 ounces with a SD of about 3.5 ounces, and that the distribution is approximately normal, to generate population data:

[41] https://github.com/genomicsclass/labs/tree/master/inference/monte_carlo.Rmd

```
controls<- rnorm(5000, mean=24, sd=3.5)
```

After we generate the data, we can then repeat the exercise above. We no longer have to use the `sample` function since we can re-generate random normal numbers. The `ttestgenerator` function therefore can be written as follows:

```
ttestgenerator <- function(n, mean=24, sd=3.5) {
  cases <- rnorm(n,mean,sd)
  controls <- rnorm(n,mean,sd)
  tstat <- (mean(cases)-mean(controls)) /
      sqrt( var(cases)/n + var(controls)/n )
  return(tstat)
  }
```

Exercises

We have used Monte Carlo simulation throughout this chapter to demonstrate statistical concepts; namely, sampling from the population. We mostly applied this to demonstrate the statistical properties related to inference on differences in averages. Here, we will consider examples of how Monte Carlo simulations are used in practice.

1. Imagine you are William_Sealy_Gosset[a] and have just mathematically derived the distribution of the t-statistic when the sample comes from a normal distribution. Unlike Gosset you have access to computers and can use them to check the results.

   Let's start by creating an outcome. Set the seed at 1, use `rnorm` to generate a random sample of size 5, $X_1, \ldots, X_5$ from a standard normal distribution, then compute the t-statistic $t = \sqrt{5}\,\bar{X}/s$ with s the sample standard deviation. What value do you observe?
2. You have just performed a Monte Carlo simulation using `rnorm` , a random number generator for normally distributed data. Gosset's mathematical calculation tells us that this random variable follows a t-distribution with $N-1$ degrees of freedom. Monte Carlo simulations can be used to check the theory: we generate many outcomes and compare them to the theoretical result. Set the seed to 1, generate $B = 1000$ t-statistics as done in exercise 1. What percent are larger than 2?
3. The answer to exercise 2 is very similar to the theoretical prediction: `1-pt(2,df=4)`. We can check several such quantiles using the `qqplot` function.

   To obtain quantiles for the t-distribution we can generate percentiles from just above 0 to just below 1: `B=100; ps = seq(1/(B+1), 1-1/(B+1),len=B)` and compute the quantiles with `qt(ps,df=4)`. Now we can use `qqplot` to compare these theoretical quantiles to those obtained in the Monte Carlo simulation. Use Monte Carlo simulation developed for exercise 2 to corroborate that the t-statistic $t = \sqrt{N}\,\bar{X}/s$ follows a t-distribution for several values of N.

For which sample sizes does the approximation best work?

- A) Larger sample sizes.
- B) Smaller sample sizes.
- C) The approximations are spot on for all sample sizes.
- D) None. We should use CLT instead.

4. Use Monte Carlo simulation to corroborate that the t-statistic comparing two means and obtained with normally distributed (mean `0` and sd) data follows a t-distribution. In this case we will use the `t.test` function with `var.equal=TRUE`. With this argument the degrees of freedom will be `df=2*N-2` with `N` the sample size. For which sample sizes does the approximation best work?
   - A) Larger sample sizes.
   - B) Smaller sample sizes.
   - C) The approximations are spot on for all sample sizes.
   - D) None. We should use CLT instead.
5. Is the following statement true or false? If instead of generating the sample with `X=rnorm(15)`, we generate it with binary data `X=rbinom(n=15,size=1,prob=0.5)` then the t-statistic

   ```
   tstat <- sqrt(15)*mean(X) / sd(X)
   ```

   is approximated by a t-distribution with 14 degrees of freedom.
6. Is the following statement true or false? If instead of generating the sample with `X=rnorm(N)` with `N=500`, we generate the data with binary data `X=rbinom(n=500,size=1,prob=0.5)`, then the t-statistic `sqrt(N)*mean(X)/sd(X)` is approximated by a t-distribution with 499 degrees of freedom.
7. We can derive approximation of the distribution of the sample average or the t-statistic theoretically. However, suppose we are interested in the distribution of a statistic for which a theoretical approximation is not immediately obvious.

   Consider the sample median as an example. Use a Monte Carlo to determine which of the following best approximates the median of a sample taken from normally distributed population with mean `0` and standard deviation `1`.
   - A) Just like for the average, the sample median is approximately normal with mean `0` and SD $1/\sqrt{N}$.
   - B) The sample median is not approximately normal.
   - C) The sample median is t-distributed for small samples and normally distributed for large ones.
   - D) The sample median is approximately normal with mean `0` and SD larger than $1/\sqrt{N}$.

[a] https://en.wikipedia.org/wiki/William_Sealy_Gosset

Permutation Tests

The R markdown document for this section is available here[42].

Suppose we have a situation in which none of the standard mathematical statistical approximations apply. We have computed a summary statistic, such as the difference in mean, but do not have a useful approximation, such as that provided by the CLT. In practice, we do not have access to all values in the population so we can't perform a simulation as done above. Permutation tests can be useful in these scenarios.

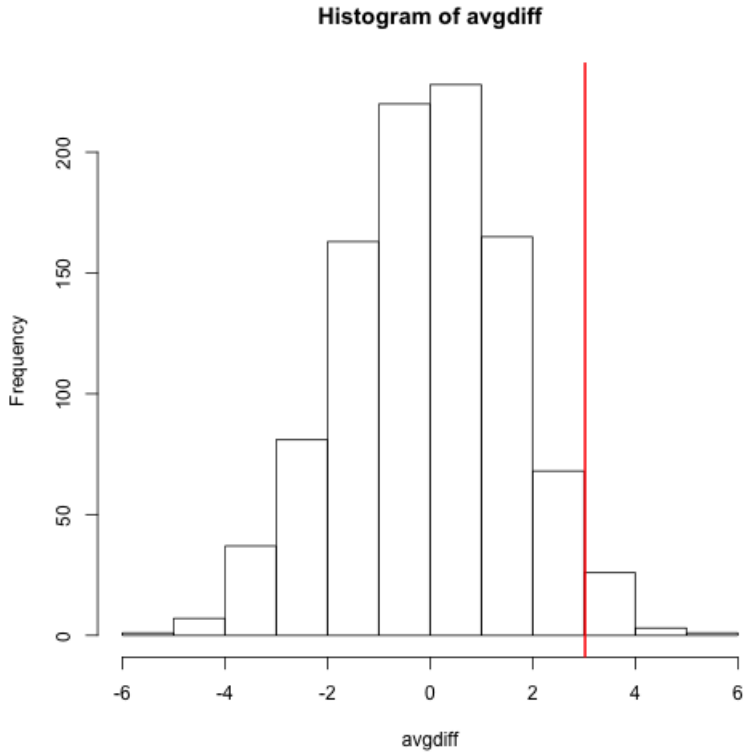
We are back to the scenario were we only have 10 measurements for each group.

```
dat=read.csv("femaleMiceWeights.csv")

library(dplyr)

control <- filter(dat,Diet=="chow") %>% select(Bodyweight) %>% unlist
treatment <- filter(dat,Diet=="hf") %>% select(Bodyweight) %>% unlist
obsdiff <- mean(treatment)-mean(control)
```

In previous sections, we showed parametric approaches that helped determine if the observed difference was significant. Permutation tests take advantage of the fact that if we randomly shuffle the cases and control labels, then the null is true. So we shuffle the cases and control labels and assume that the ensuing distribution approximates the null distribution. Here is how we generate a null distribution by shuffling the data 1,000 times:

```
N <- 12
avgdiff <- replicate(1000, {
    all <- sample(c(control,treatment))
    newcontrols <- all[1:N]
    newtreatments <- all[(N+1):(2*N)]
  return(mean(newtreatments) - mean(newcontrols))
})
hist(avgdiff)
abline(v=obsdiff, col="red", lwd=2)
```

[42] https://github.com/genomicsclass/labs/tree/master/inference/permutation_tests.Rmd



Histogram of difference between averages from permutations. Vertical line shows the observed difference.

How many of the null means are bigger than the observed value? That proportion would be the p-value for the null. We add a 1 to the numerator and denominator to account for misestimation of the p-value (for more details see Phipson and Smyth, Permutation P-values should never be zero[43]).

```
#the proportion of permutations with larger difference
(sum(abs(avgdiff) > abs(obsdiff)) + 1) / (length(avgdiff) + 1)
```

```
## [1] 0.07392607
```

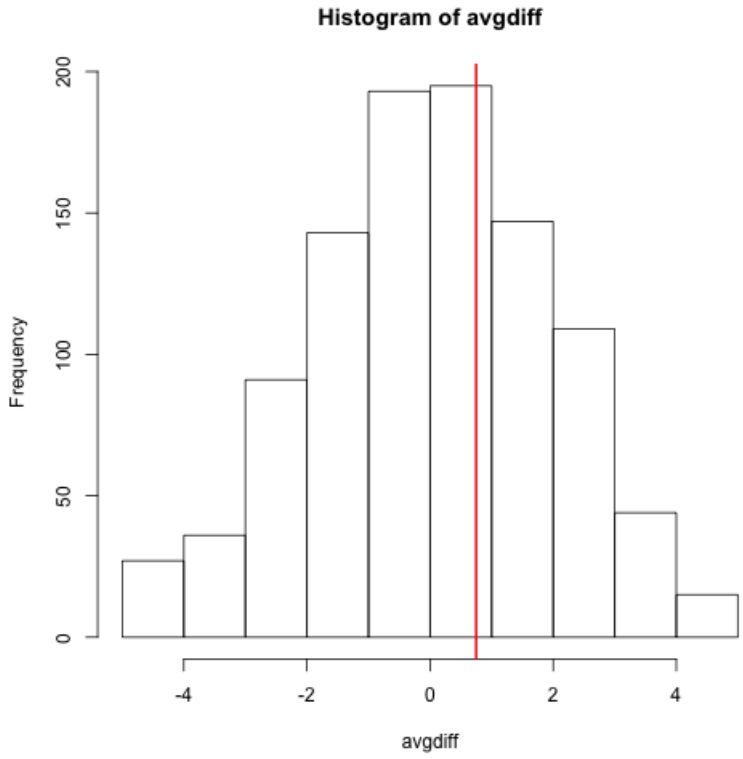
Now let's repeat this experiment for a smaller dataset. We create a smaller dataset by sampling:

```
N <- 5
control <- sample(control,N)
treatment <- sample(treatment,N)
obsdiff <- mean(treatment)- mean(control)
```

and repeat the exercise:

[43] http://www.ncbi.nlm.nih.gov/pubmed/21044043

```
avgdiff <- replicate(1000, {
    all <- sample(c(control,treatment))
    newcontrols <- all[1:N]
    newtreatments <- all[(N+1):(2*N)]
  return(mean(newtreatments) - mean(newcontrols))
})
hist(avgdiff)
abline(v=obsdiff, col="red", lwd=2)
```



Histogram of difference between averages from permutations for smaller sample size. Vertical line shows the observed difference.

Now the observed difference is not significant using this approach. Keep in mind that there is no theoretical guarantee that the null distribution estimated from permutations approximates the actual null distribution. For example, if there is a real difference between the populations, some of the permutations will be unbalanced and will contain some samples that explain this difference. This implies that the null distribution created with permutations will have larger tails than the actual null distribution. This is why permutations result in conservative p-values. For this reason, when we have few samples, we can't do permutations.

Note also that permutations tests still have assumptions: samples are assumed to be independent and "exchangeable". If there is hidden structure in your data, then permutation tests can result

in estimated null distributions that underestimate the size of tails because the permutations may destroy the existing structure in the original data.

Exercises

We will use the following dataset to demonstrate the use of permutations:

```
url <- "https://raw.githubusercontent.com/genomicsclass/dagdata/master/inst/extd\
ata/babies.txt"
filename <- basename(url)
download(url, destfile=filename)
babies <- read.table("babies.txt", header=TRUE)
bwt.nonsmoke <- filter(babies, smoke==0) %>% select(bwt) %>% unlist
bwt.smoke <- filter(babies, smoke==1) %>% select(bwt) %>% unlist
```

1. We will generate the following random variable based on a sample size of 10 and observe the following difference:

   ```
   N=10
   set.seed(1)
   nonsmokers <- sample(bwt.nonsmoke , N)
   smokers <- sample(bwt.smoke , N)
   obs <- mean(smokers) - mean(nonsmokers)
   ```

   The question is whether this observed difference is statistically significant. We do not want to rely on the assumptions needed for the normal or t-distribution approximations to hold, so instead we will use permutations. We will reshuffle the data and recompute the mean. We can create one permuted sample with the following code:

   ```
   dat <- c(smokers,nonsmokers)
   shuffle <- sample( dat )
   smokersstar <- shuffle[1:N]
   nonsmokersstar <- shuffle[(N+1):(2*N)]
   mean(smokersstar)-mean(nonsmokersstar)
   ```

   The last value is one observation from the null distribution we will construct. Set the seed at 1, and then repeat the permutation 1,000 times to create a null distribution. What is the permutation derived p-value for our observation?
2. Repeat the above exercise, but instead of the differences in mean, consider the differences in median `obs <- median(smokers) - median(nonsmokers)`. What is the permutation based p-value?

Association Tests

The R markdown document for this section is available here[44].

The statistical tests we have covered up to now leave out a substantial portion of life science projects. Specifically, we are referring to data that is binary, categorical and ordinal. To give a very specific example, consider genetic data where you have two groups of genotypes (AA/Aa or aa) for cases and controls for a given disease. The statistical question is if genotype and disease are associated. As in the examples we have been studying previously, we have two populations (AA/Aa and aa) and then numeric data for each, where disease status can be coded as 0 or 1. So why can't we perform a t-test? Note that the data is either 0 (control) or 1 (cases). It is pretty clear that this data is not normally distributed so the t-distribution approximation is certainly out of the question. We could use CLT if the sample size is large enough; otherwise, we can use *association tests*.

Lady Tasting Tea

One of the most famous examples of hypothesis testing was performed by R.A. Fisher[45]. An acquaintance of Fisher's claimed that she could tell if milk was added before or after tea was poured. Fisher gave her four pairs of cups of tea: one with milk poured first, the other after. The order was randomized. Say she picked 3 out 4 correctly, do we believe she has a special ability? Hypothesis testing helps answer this question by quantifying what happens by chance. This example is called the "Lady tasting tea" experiment (and, as it turns out, Fisher's friend was a scientist herself, Muriel Bristol[46]).

The basic question we ask is: if the tester is actually guessing, what are the chances that she gets 3 or more correct? Just as we have done before, we can compute a probability under the null hypothesis that she is guessing four of each. If we assume this null hypothesis, we can think of this particular examples as picking 4 balls out of an urn with 4 green (correct answer) and 4 red (incorrect answer) balls.

Under the null hypothesis that she is simply guessing, each ball has the same chance of being picked. We can then use combinatorics to figure out each probability. The probability of picking 3 is $\binom{4}{3}\binom{4}{1}/\binom{8}{4} = 16/70$. The probability of picking all 4 correct is $\binom{4}{4}\binom{4}{0}/\binom{8}{4} = 1/70$. Thus, the chance of observing a 3 or something more extreme, under the null hypothesis, is ≈ 0.24. This is the p-value. The procedure that produced this p-value is called *Fisher's exact test* and it uses the *hypergeometric distribution*.

Two By Two Tables

The data from the experiment above can be summarized by a 2 by 2 table:

[44] https://github.com/genomicsclass/labs/tree/master/inference/association_tests.Rmd

[45] https://en.wikipedia.org/wiki/Ronald_Fisher

[46] https://en.wikipedia.org/wiki/Muriel_Bristol

```
tab <- matrix(c(3,1,1,3),2,2)
rownames(tab)<-c("Poured Before","Poured After")
colnames(tab)<-c("Guessed before","Guessed after")
tab
```

```
##               Guessed before Guessed after
## Poured Before              3             1
## Poured After               1             3
```

The function `fisher.test` performs the calculations above and can be obtained like this:

```
fisher.test(tab,alternative="greater")
```

```
##
##         Fisher's Exact Test for Count Data
##
## data:  tab
## p-value = 0.2429
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.3135693       Inf
## sample estimates:
## odds ratio
##   6.408309
```

Chi-square Test

Genome-wide association studies (GWAS) have become ubiquitous in biology. One of the main statistical summaries used in these studies are Manhattan plots. The y-axis of a Manhattan plot typically represents the negative of log (base 10) of the p-values obtained for association tests applied at millions of single nucleotide polymorphisms (SNP). The x-axis is typically organized by chromosome (chromosome 1 to 22, X, Y, etc.). These p-values are obtained in a similar way to the test performed on the tea taster. However, in that example the number of green and red balls is experimentally fixed and the number of answers given for each category is also fixed. Another way to say this is that the sum of the rows and the sum of the columns are fixed. This defines constraints on the possible ways we can fill the 2 by 2 table and also permits us to use the hypergeometric distribution. In general, this is not the case. Nonetheless, there is another approach, the Chi-squared test, which is described below.

Imagine we have 250 individuals, where some of them have a given disease and the rest do not. We observe that 20% of the individuals that are homozygous for the minor allele (aa) have the disease compared to 10% of the rest. Would we see this again if we picked another 250 individuals?

Let's create a dataset with these percentages:

```
disease=factor(c(rep(0,180),rep(1,20),rep(0,40),rep(1,10)),
               labels=c("control","cases"))
genotype=factor(c(rep("AA/Aa",200),rep("aa",50)),
                levels=c("AA/Aa","aa"))
dat <- data.frame(disease, genotype)
dat <- dat[sample(nrow(dat)),] #shuffle them up
head(dat)
```

```
##     disease genotype
## 67  control    AA/Aa
## 93  control    AA/Aa
## 143 control    AA/Aa
## 225 control       aa
## 50  control    AA/Aa
## 221 control       aa
```

To create the appropriate two by two table, we will use the function `table`. This function tabulates the frequency of each level in a factor. For example:

```
table(genotype)
```

```
## genotype
## AA/Aa    aa
##   200    50
```

```
table(disease)
```

```
## disease
## control   cases
##     220      30
```

If you you feed the function two factors, it will tabulate all possible pairs and thus create the two by two table:

```
tab <- table(genotype,disease)
tab
```

```
##         disease
## genotype control cases
##    AA/Aa     180    20
##    aa         40    10
```

Note that you can feed `table` n factors and it will tabulate all n-tables.

The typical statistics we use to summarize these results is the odds ratio (OR). We compute the odds of having the disease if you are an "aa": 10/40, the odds of having the disease if you are an "AA/Aa": 20/180, and take the ratio: $(10/40)/(20/180)$

```
(tab[2,2]/tab[2,1]) / (tab[1,2]/tab[1,1])
```

```
## [1] 2.25
```

To compute a p-value, we don't use the OR directly. We instead assume that there is no association between genotype and disease, and then compute what we expect to see in each *cell* of the table (note: this use of the word "cell" refers to elements in a matrix or table and has nothing to do with biological cells). Under the null hypothesis, the group with 200 individuals and the group with 50 individuals were each randomly assigned the disease with the same probability. If this is the case, then the probability of disease is:

```
p=mean(disease=="cases")
p
```

```
## [1] 0.12
```

The expected table is therefore:

```
expected <- rbind(c(1-p,p)*sum(genotype=="AA/Aa"),
                  c(1-p,p)*sum(genotype=="aa"))
dimnames(expected)<-dimnames(tab)
expected
```

```
##         disease
## genotype control cases
##    AA/Aa     176    24
##    aa         44     6
```

The Chi-square test uses an asymptotic result (similar to the CLT) related to the sums of independent binary outcomes. Using this approximation, we can compute the probability of seeing a deviation from the expected table as big as the one we saw. The p-value for this table is:

```
chisq.test(tab)$p.value
```

```
## [1] 0.08857435
```

Large Samples, Small p-values

As mentioned earlier, reporting only p-values is not an appropriate way to report the results of your experiment. Many genetic association studies seem to over emphasize p-values. They have large sample sizes and report impressively small p-values. Yet when one looks closely at the results, we realize odds ratios are quite modest: barely bigger than 1. In this case the difference of having genotype AA/Aa or aa might not change an individual's risk for a disease in an amount which is *practically significant*, in that one might not change one's behavior based on the small increase in risk.

There is not a one-to-one relationship between the odds ratio and the p-value. To demonstrate, we recalculate the p-value keeping all the proportions identical, but increasing the sample size by 10, which reduces the p-value substantially (as we saw with the t-test under the alternative hypothesis):

```
tab<-tab*10
chisq.test(tab)$p.value
```

```
## [1] 1.219624e-09
```

Confidence Intervals For The Odd Ratio

Computing confidence intervals for the OR is not mathematically straightforward. Unlike other statistics, for which we can derive useful approximations of their distributions, the OR is not only a ratio, but a ratio of ratios. Therefore, there is no simple way of using, for example, the CLT.

One approach is to use the theory of *generalized linear models* which provides estimates of the log odds ratio, rather than the OR itself, that can be shown to be asymptotically normal. Here we provide R code without presenting the theoretical details (for further details please see a reference on generalized linear models such as Wikipedia[47] or McCullagh and Nelder, 1989[48]):

[47] https://en.wikipedia.org/wiki/Generalized_linear_model

[48] https://books.google.com/books?hl=en&lr=&id=h9kFH2_FfBkC

```
fit <- glm(disease~genotype,family="binomial",data=dat)
coeftab<- summary(fit)$coef
coeftab
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -2.1972246  0.2356828 -9.322803 1.133070e-20
## genotypeaa   0.8109302  0.4249074  1.908487 5.632834e-02
```

The second row of the table shown above gives you the estimate and SE of the log odds ratio. Mathematical theory tells us the this estimate is approximately normally distributed. We can therefore form a confidence interval and then exponentiate to provide a confidence interval for the OR.

```
ci <- coeftab[2,1] + c(-2,2)*coeftab[2,2]
exp(ci)
```

```
## [1] 0.9618616 5.2632310
```

The confidence includes 1, which is consistent with the p-value being bigger than 0.05. Note that the p-value shown here is based on a different approximation to the one used by the Chi-square test, which is why they differ.

Exercises

We showed how to calculate a Chi-square test from a table. Here we will show how to generate the table from data which is in the form of a dataframe, so that you can then perform an association test to see if two columns have an enrichment (or depletion) of shared occurrences.

Download the https://studio.edx.org/c4x/HarvardX/PH525.1x/asset/assoctest.csv[a] file into your R working directory, and then read it into R: `d = read.csv("assoctest.csv")`

1. This dataframe reflects the allele status (either AA/Aa or aa) and the case/control status for 72 individuals. Compute the Chi-square test for the association of genotype with case/control status (using the `table` function and the `chisq.test` function). Examine the table to see if there appears to be an association. What is the X-squared statistic?
2. Compute Fisher's exact test `fisher.test` for the same table. What is the p-value?

[a] assoctest.csv